

CHANGE-POINT DETECTION IN DYNAMICAL SYSTEMS USING AUTO-ASSOCIATIVE NEURAL NETWORKS

by

Meshack Linda Bulunga

Thesis submitted in partial fulfillment
of the requirements for the Degree

of

MASTER OF SCIENCE IN ENGINEERING
(CHEMICAL ENGINEERING)



in the Faculty of Engineering
at Stellenbosch University

Supervised by
Prof. Chris Aldrich

March 2012

DECLARATION

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

.....
Signature

.....
Date

ABSTRACT

In this research work, auto-associative neural networks have been used for change-point detection. This is a nonlinear technique that employs the use of artificial neural networks as inspired among other by Frank Rosenblatt's linear perceptron algorithm for classification. An auto-associative neural network was used successfully to detect change-points for various types of time series data. Its performance was compared to that of singular spectrum analysis developed by Moskvina and Zhigljavsky. Fraction of Explained Variance (FEV) was also used to compare the performance of the two methods. FEV indicators are similar to the eigenvalues of the covariance matrix in principal component analysis.

Two types of time series data were used for change-point detection: Gaussian data series and nonlinear reaction data series. The Gaussian data had four series with different types of change-points, namely a change in the mean value of the time series (T1), a change in the variance of the time series (T2), a change in the autocorrelation of the time series (T3), and a change in the crosscorrelation of two time series (T4). Both linear and nonlinear methods were able to detect the changes for T1, T2 and T4. None of them could detect the changes in T3. With the Gaussian data series, linear singular spectrum analysis (LSSA) performed as well as the NLSSA for the change point detection. This is because the time series was linear and the nonlinearity of the NLSSA was therefore not important. LSSA did even better than NLSSA when comparing FEV values, since it is not subject to suboptimal solutions as could sometimes be the case with autoassociative neural networks.

The nonlinear data consisted of the Belousov-Zhabotinsky (BZ) reactions, autocatalytic reaction time series data and data representing a predator-prey system. With the NLSSA methods, change points could be detected accurately in all three systems, while LSSA only managed to detect the change-point on the BZ reactions and the predator-prey system. The NLSSA method also fared better than the LSSA method when comparing FEV values for the BZ reactions. The LSSA method was able to model the autocatalytic reactions fairly accurately, being able to explain 99% of the variance in the data with one component only. NLSSA with two nodes on the bottleneck attained an FEV of 87%. The performance of both NLSSA and LSSA were comparable for the predator-prey system, both systems, where both could

attain FEV values of 92% with a single component. An auto-associative neural network is a good technique for change point detection in nonlinear time series data. However, it offers no advantage over linear techniques when the time series data are linear.

OPSOMMING

In hierdie navorsing is outoassosiatiewe neurale netwerk gebruik vir veranderingspuntwaarneming. Dis is 'n nielineêre tegniek wat neurale netwerke gebruik soos onder andere geïnspireer deur Frank Rosenblatt se lineêre perseptron algoritme vir klassifikasie. 'n Outoassosiatiewe neurale netwerk is suksesvol gebruik om veranderingspunte op te spoor in verskeie tipes tydreeksdata. Die prestasie van die outoassosiatiewe neurale netwerk is vergelyk met singuliere spektrale ontleding soos ontwikkel deur Moskvina en Zhigljavsky. Die fraksie van die verklaarde variansie (FEV) is ook gebruik om die prestasie van die twee metodes te vergelyk. FEV indikatore is soortgelyk aan die eiewaardes van die kovariansiematriks in hoofkomponentontleding.

Twee tipes tydreeksdata is gebruik vir veranderingspuntopsporing: Gaussiaanse tydreeksse en nielineêre reaksiedatareeksse. Die Gaussiaanse data het vier reekse gehad met verskillende veranderingspunte, naamlik 'n verandering in die gemiddelde van die tydreeksdata (T1), 'n verandering in die variansie van die tydreeksdata (T2), 'n verandering in die outokorrelasie van die tydreeksdata (T3), en 'n verandering in die kruiskorrelasie van twee tydreeksse (T4). Beide lineêre en nielineêre metodes kon die veranderinge in T1, T2 en T4 opspoor. Nie een het egter daarin geslaag om die verandering in T3 op te spoor nie. Met die Gaussiaanse tydreeks het lineêre singuliere spektrumanalise (LSSA) net so goed gevaar soos die outoassosiatiewe neurale netwerk of nielineêre singuliere spektrumanalise (NLSSA), aangesien die tydreeksse lineêr was en die vermoë van die NLSSA metode om nielineêre gedrag te identifiseer dus nie belangrik was nie. Inteendeel, die LSSA metode het 'n groter FEV waarde getoon as die NLSSA metode, omdat LSSA ook nie blootgestel is aan suboptimale oplossings, soos wat soms die geval kan wees met die afrigting van die outoassosiatiewe neural netwerk nie.

Die nielineêre data het bestaan uit die Belousov-Zhabotinsky (BZ) reaksiedata, 'n outokatalitiese reaksietydreeksdata en data wat 'n roofdier-prooistelsel verteenwoordig het. Met die NLSSA metode kon veranderingspunte betroubaar opgespoor word in al drie tydreeksse, terwyl die LSSA metode net die veranderingspunt in die BZ reaksie en die roofdier-prooistelsel kon opspoor. Die NLSSA metode het ook beter gevaar as die LSSA metode wanneer die FEV waardes vir die BZ reaksies vergelyk word. Die LSSA metode kon die outokatalitiese reaksies redelik akkuraat modelleer, en kon met slegs een komponent 99% van die

variansie in die data verklaar. Die NLSSA metode, met twee nodes in sy bottelneklaag, kon 'n FEV waarde van slegs 87% behaal. Die prestasie van beide metodes was vergelykbaar vir die roofdier-prooidata, met beide wat FEV waardes van 92% kon behaal met hulle een komponent. 'n Outoassosiatiewe neural netwerk is 'n goeie metode vir die opspoor van veranderingspunte in nielineêre tydreeksdata. Dit hou egter geen voordeel in wanneer die data lineêr is nie.

ACKNOWLEDGEMENTS

My thanks go to the University of Stellenbosch's Department of Chemical Engineering for the opportunity to be involved in the department's research. I extend my most deep felt gratitude to Professor Chris Aldrich for his support, guidance and supervision; Dr Gordon Jemwa for his assistance and advice; Callisto Kazembe for being a friend when I needed one. Thanks to the NRF for the funding, without it this work would have not been possible. Thanks to office A601 inhabitants; you all contributed in different ways to my work.

Lastly but not least to my family; Nelly – thanks for everything, it means a lot to me. Without your support life would have been difficult. To my kids, I love you girls.

TABLE OF CONTENTS

ABSTRACT	iv
OPSOMMING	vi
LIST OF ABBREVIATIONS	xi
LIST OF SYMBOLS	xii
Chapter 1 Introduction	1
1.1 Process Monitoring	2
1.2 Model-Based Process Monitoring Techniques	3
1.3 Data-Driven Methods.....	5
1.4 Problem Statement.....	7
1.5 Research Objectives	7
1.6 Organization of Thesis.....	7
Chapter 2 Change-Point Detection: A Literature Review	8
Chapter 3 Change-Point Detection with SSA and Artificial Neural Networks.....	15
3.1 Singular Spectrum Analysis (SSA)	15
3.2 Change-Point Detection with Singular Spectrum Analysis.....	21
3.2.1 Algorithm.....	21
3.2.2 Choice of Parameters	23
3.3 Neural Networks	24
3.3.1 Multilayer Perceptron (MLP)	25
3.3.2 Data Pre-processing	27
3.3.3 Limitations of Neural Networks	28
3.3.4 Auto-associative Neural Networks (NLSSA/NLPCA)	31
3.3.5 Feature Extraction	33
3.4 Nonlinear SSA with Auto-Associative Neural Networks	35
3.4.1 Methodology for Change-Point Detection Using NLSSA	37
Chapter 4 Case Studies.....	46
4.1 Gaussian Time Series Data.....	46
4.2 Belousov-Zhabotinsky (BZ) reactions	47
4.3 Autocatalytic Process	48
4.4 Lotka-Volterra Model.....	50
Chapter 5 Results and Discussion	53
5.1 Gaussian Time Series Data.....	53
5.1.1 Network Training.....	56
5.1.2 Change-Point Detection of Gaussian Data (T1 and T2).....	57

5.2 Autoregressive Process	61
5.2.1 Network Training.....	62
5.2.2 Change-Point Detection Using Nonlinear Singular Spectrum Analysis.....	64
5.2.3 Change-Point Detection Using Linear Singular Spectrum Analysis	65
5.2.4 Comparison between NLSSA and LSSA Using FEV	66
5.3 Cross-correlated Time Series (T4)	67
5.3.1 Network Training.....	68
5.3.2 Change-Point Detection with NLSSA.....	68
5.3.3 Network Training for Multicorrelated Time Series	70
5.3.4 Change-Point Detection with NLSSA.....	71
5.3.5 Change-Point Detection with LSSA	72
5.4 Belousov-Zhabotinsky (BZ) reactions	72
5.4.1 Calculation of Embedding Parameters.....	73
5.4.2 Network Training.....	74
5.4.3 Change-Point Detection with NLSSA.....	77
5.4.4 Change-Point Detection with LSSA	77
5.4.5 Comparison between NLSSA and LSSA	78
5.4.6 Feature Extraction with NLSSA	78
5.5 Autocatalytic Reaction System	80
5.5.1 Calculation of Embedding Parameters.....	80
5.5.2 Network Training.....	82
5.5.3 Change-Point Detection with NLSSA.....	84
5.5.4 Change-Point Detection with LSSA	84
5.5.5 Comparison between NLSSA and LSSA	85
5.5.6 Feature Extraction with NLSSA	86
5.6 Predator-Prey.....	87
5.6.1 Calculation of Embedding Parameters.....	88
5.6.2 Network Training.....	90
5.6.3 Change-Point Detection for Predator-Prey System with NLSSA	92
5.6.4 Change-Point Detection for a Predator-Prey System with LSSA.....	94
5.6.5 Comparison between NLSSA and LSSA	94
5.7 Comparative Analysis of Change-Point Detection Techniques.....	95
5.8 Practical Considerations	96
Chapter 6 Conclusions.....	97
Appendices.....	99
A. CUSUM Algorithm.....	99
B. Estimation of Embedding Parameters.....	103
References.....	109

LIST OF ABBREVIATIONS

ACF	autocorrelation function
AMI	average mutual information
ANN	artificial neural network
CDA	confirmatory data analysis
CMAC	cerebellar model articulation controller
CUSUM	cumulative summation
DCD	distributed change-point detection
EDA	exploratory data analysis
EWMA	exponentially weighted moving average
FEV	fraction of explained variance
FNN	false nearest neighbour
LM	Levenberg-Marquardt
LSSA	linear singular spectrum analysis
MLP	multilayer perceptron
MPC	model predictive control
MSE	mean square error
MSW	mean square weight
NLPCA	nonlinear principal component analysis
NLSSA	nonlinear singular spectrum analysis
ODE	ordinary differential equation
PC	principal component
PCA	principal component analysis
RBFN	radial bases function networks
SIC	Schwartz information criterion
SPC	statistical process control
SSA	singular spectrum analysis

LIST OF SYMBOLS

τ	time delay
m	embedding dimension
$I(\tau)$	average mutual information at time lag(t)
λ	eigenvalue
θ_i	bias associated to the unit i
w_{ij}	network input weights associated to the unit i
$g(x)$	activation function
σ^2	variance
α	confidence level
M_{FF}	model structure
D_M	set of possible model parameters
μ	mean
S_k	log –likelihood ratio
D_n	squared distances
v_n	estimator of the normalized sum of squared distances
γ	performance ratio
\tilde{x}_i^n	rescaled variables that have zero mean and unit standard deviation

Chapter 1 Introduction

In the chemical industry and other related industries, there has been a large move to produce higher quality products, reduce product rejection rates, and satisfy increasingly stringent safety and environmental regulations. Process operations once considered acceptable are no longer adequate (Russell et al., 2000). To meet the higher standards, modern chemical processes contain a large number of variables operating under closed loop control. The standard process controllers, such as PID controllers and model predictive controllers, are designed to maintain satisfactory operations by compensating for the effects of disturbances and changes occurring in the process (Russell et al., 2000).

Besides the addition of PID controllers, numerous variables are measured and recorded at many time points. The resulting process data is highly correlated and is subject to considerable noise. In the absence of an appropriate method for processing such data, only limited information can be extracted and this leads to poor understanding of the process, in turn resulting in an unstable operation. However, if properly processed, the abundance of process data can provide a wealth of information, enabling good understanding of the process and improved monitoring of the process. Process monitoring plays an important role in the analysis and interpretation of plant data. It helps process operators to make informed decisions when operating.

The field of process control has made considerable progress in the last three decades with the advent of computer control of complex processes (Venkatasubramanian et al., 2003). Regulatory control is now performed in an automated manner with the aid of computers. However, managing process plants still remains largely a manual activity, executed by process controllers. This involves the timely detection of an abnormal event, diagnosing its origins and then taking appropriate supervisory control decisions and actions to bring the process back to a normal, safe, operating state. The size and complexity of modern process plants make it extremely difficult for the process controllers to timeously detect and diagnose a variety of malfunctions. This is further aggravated by the fact that in large process plants there may be as many as 1500 process variables observed every few

seconds (Venkatasubramanian et al., 2003). It is therefore of no surprise that human operators tend to contribute to industrial accidents.

Industrial statistics show that about 70% of industrial accidents are caused by human errors (Venkatasubramanian et al., 2003). Recent events have shown that large scale plant accidents are not just a thing of the past; two of the worst ever chemical plant accidents happened as recently as the 1980's, namely Union Carbide's Bhopal, India accident and Occidental Petroleum's Piper Alpha Accident. Such catastrophes have significant impact on safety, environment and the economy (Venkatasubramanian, 2003). The explosion at the Kuwait Petrochemical's Mina Al-Ahmedhi refinery in June 2000 resulted in damages worth about \$400 million, and the explosion of the offshore oil platform of Petrobras, Brazil and its subsequent sinking into the sea in March 2001 resulted in losses estimated to be about \$5 billion (Venkatasubramanian, 2003).

Industrial statistics also show that even though the occurrence of major industrial accidents may not be common, minor accidents are very frequent and occur almost daily resulting in many occupational injuries and sickness, costing billions of dollars every year (Venkatasubramanian et al., 2003). This shows that there is still a lot that needs to be done to assist the performance of human operators, to enhance their diagnostic capability and consequently, enable better judgments and decisions. This will help to reduce the number of incidents and health related losses, improving safety. Process monitoring techniques are one of the aspects that can be explored to help equip the human process operators.

1.1 Process Monitoring

Process monitoring is a continuous real-time task of recognizing anomalies in the behavior of a dynamic system. It is a means of detecting unwanted process disturbances on time, and might be in the form of: fault detection and diagnosis (monitoring), condition-based maintenance of industrial processes, safety of complex systems (aircrafts, boats, rockets, nuclear power plants, chemical technological processes, etc.), quality control, prediction of natural catastrophic events (earthquakes, tsunamis, etc.), or monitoring in biomedicine (Basseville et al., 1993). The common feature of process monitoring is the detection of one or more abrupt changes in characteristic properties of the considered object. The key difficulty is to

detect intrinsic changes that are not necessarily directly observed, and that are measured together with other types of disturbances.

Many monitoring problems can be stated as the problem of detecting a change in the parameters of a static or dynamic stochastic system. In the last two decades there has been an increase in the search for more efficient methods that can be used to analyze process data and hence improve process monitoring. The monitoring of chemical processes and the diagnosis of faults in these processes are very important aspects of process systems engineering because they are integral to the successful execution of planned operations and to improving process productivity and product quality (Lee et al., 2004).

Traditionally, statistical process control (SPC) charts such as Shewhart, cumulative sum (CUSUM) and exponentially weighted moving average (EWMA) charts have been used to monitor processes and improve product quality. However, such univariate control charts show poor detection performance when applied to multivariate processes (Lee et al., 2004). With advances in computer technology, process monitoring techniques have also improved, and process models can be built which run in parallel with the actual process plant. The output of the model is compared with the real plant output. Any deviation between what the model predicts and what the actual plant produces indicates an abnormality.

Process monitoring techniques can be classified into two categories: model based process monitoring and process history based methods.

1.2 Model-Based Process Monitoring Techniques

A model can be defined as a representation of the essential aspects of an existing system (or a system to be constructed) which presents knowledge of that system in a usable form. This means that a model is always a simplified representation of the real system. Such a representation can provide insight into the behavior of the system, although this does not necessarily mean that this insight is physical. There are three main types of models: white box models, black box models and grey box models.

White box models (also called first principles or mechanistic models) are models that are entirely based on physical and chemical laws (thermodynamics and

continuity equations). They are as close to a full description of the real device as possible. These models give a physical insight of the system and can even be built when the system is not yet constructed (Van Lith, 2002).

Usually a set of (partial) differential equations, supplemented with algebraic equations, is used to give a mathematical description of the model. The effort needed to build these models is usually high, especially for complex chemical systems. Due to their complex nature, they end up being the slowest running type of model and require fast computers with large amounts of RAM.

Black box models (also called empirical models) do not reflect the physical structure of the system; rather, they give an input/output relation of the process. These models are useful if a physical understanding of the system is absent or not relevant for the purpose of the model. Black box models usually consist of a set of rules and equations, they are easy to optimize, and can run very rapidly (Van Lith, 2002). Mathematical descriptions used include autoregressive models (such as ARMA and ARMAX models), and artificial neural networks. Black box models are relatively simple and do not require a great deal of computing power.

Grey box models usually arise due to incomplete knowledge of the process. This results in models that use both first principles and empirical modeling strategies (Van Lith, 2002). A grey box model provides flexibility, allowing the physical layout to be altered by simply re-drawing it. A grey box model is physically a close representation of the device being modeled. This makes the model easier to work with, because you can visualize and see any changes made to the model. Because a grey box model is more generic, it cannot be as optimized as a black box model, and hence will tend to run slower and usually require more computer memory than a black box model. Tracking errors is a more involved task in a grey box model because there are more places for errors to originate.

Although the most reliable approach to process monitoring will be the use of precise first principle models, such models are not available for most processes and modeling of a complex industrial process is very difficult and time-consuming. Most model-based process monitoring methods are residual-based. In this approach, a mathematical model is created by knowing the input and the output of the system. This model is used to compare the actual output with those nominal behaviors

produced by the model, and therefore residuals are formed. At this point, a decision needs to be made on whether a deviation has occurred or not.

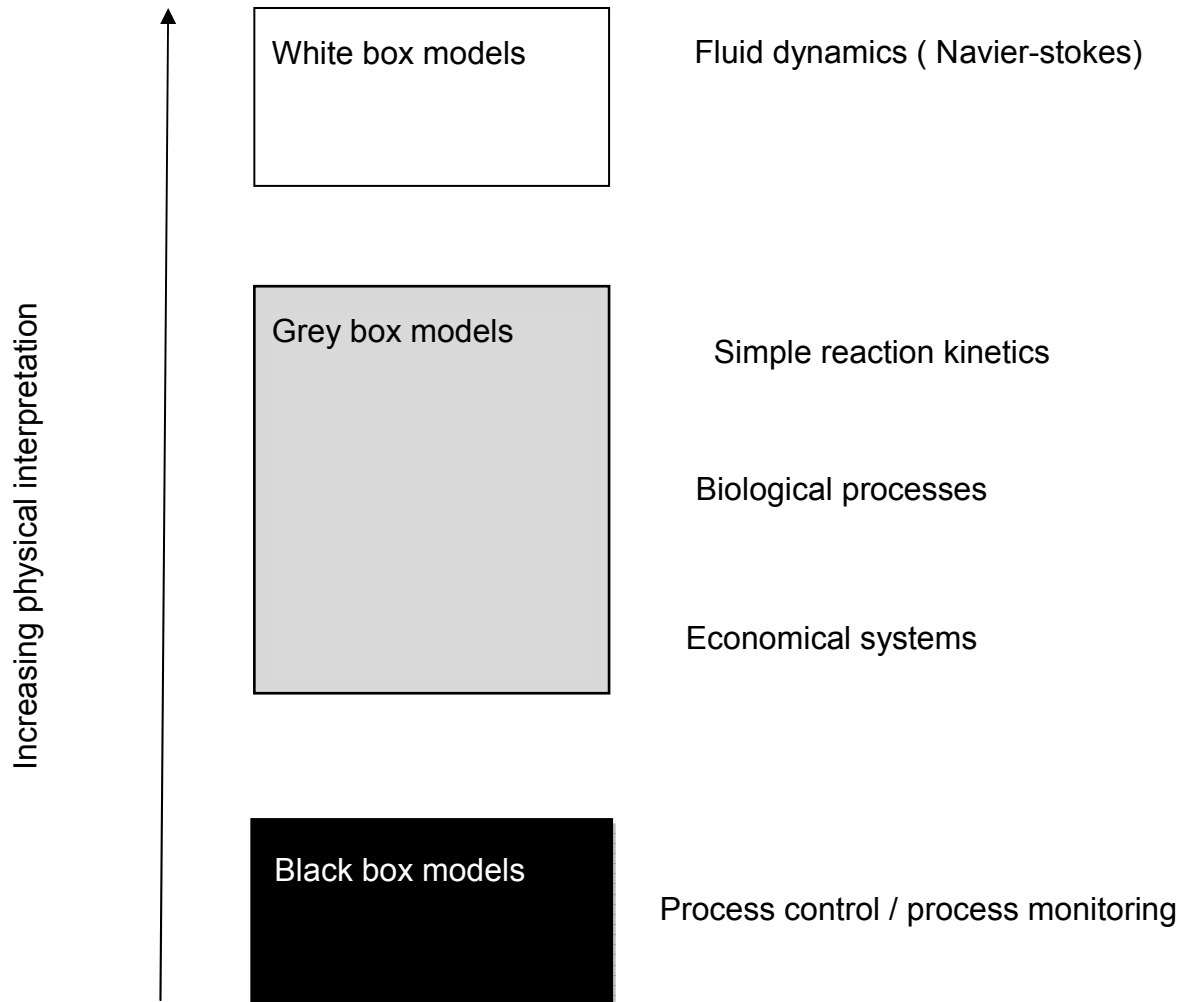


Figure 1.1: Physical interpretability of grey box models and examples of applications taken from (Van Lith, 2002).

1.3 Data-Driven Methods

Data-driven or process history-based methods do not need *apriori* knowledge of the process, only a large amount of historical process data. The historical data is transformed and presented as a priori knowledge and used to diagnose the process. Different techniques are used to transform the process history data into priori knowledge. The process of transforming the historical data into apriori knowledge is called feature extraction. This extraction process can be either qualitative or

quantitative in nature. Two of the popular methods that extract qualitative history information are the expert systems and trend modeling methods. Methods that extract quantitative information can be broadly classified as non-statistical or statistical methods (Venkatasubramanian et al., 2003).

Neural networks are an important class of non-statistical classifiers. Principal component analysis (PCA), partial least squares (PLS) and statistical pattern classifiers form a major component of statistical feature extraction methods.

Most process data is presented in the form of time series. A time series is a sequence of data points, typically measured at successive times spaced at uniform time intervals. Time series analysis comprises methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data. Time series models make use of the natural one-way ordering of time so that values for a given period will be expressed as deriving in some way from past values, rather than from future values (Hajizabath et al., 2010). Most monitoring schemes do not emphasize the exact time at which the abnormal event took place. This work focuses on detecting the exact time at which the abnormality took place as this helps in identifying the cause of the abnormality.

The problem of discovering time points at which properties of time series data change is attracting a lot attention in the data mining community (Kawahara & Sugiyama, 2009; Basseville & Nikiforov, 1993). This problem is referred to as change-point detection, or event detection, and covers a broad range of real-world problems such as fraud detection in cellular systems (Kawahara & Sugiyama, 2009), intrusion detection in computer networks (Chen et al., 2007), fault detection in engineering systems (Xie, 2005), robotics, process control, finance, EEG analysis, DNA segmentation, econometrics, disease demographics, pharmacology, psychology, geology, meteorology, and environmental studies (Frisén, 2003). The problem of identifying change-points is one of the most challenging statistical problems since both the number of change-points and their locations are unknown (Cheon & Kim, 2009).

1.4 Problem Statement

Many chemical and metallurgical processes are characterized by highly nonlinear and complex dynamics, with long time constants and significant delays. The presence of nonlinearities gives rise to structural kinetic instabilities. A lot of research has been done on nonlinear process monitoring techniques over the past two decades. This is driven by the fact that most processes are nonlinear and the methods used to model, monitor and control them are predominantly linear. Although the linear methods do model and monitor the processes fairly accurately, there are instances where they fail to capture the nonlinearity of the process. A couple of nonlinear methods have been developed and some are already being used in industry. There is no single method or technique that has all the desirable features to accurately model and monitor all processes, hence the continual need to find more and better process monitoring techniques.

1.5 Research Objectives

The main objectives of the study are:

1. Review existing change-point detection techniques proposed in literature.
2. Develop a methodology for a change-point detection technique using auto-associative neural networks and singular spectrum analysis.
3. Evaluate the performance of the proposed nonlinear change-point detection technique in detecting changes in data generated from simulated deterministic and stochastic systems by comparing it with linear singular spectrum analysis.

1.6 Organization of Thesis

Chapter two focuses on change-point detection literature survey. Chapter three focuses mainly on the theory of Singular Spectrum analysis and its change-point detection algorithm, artificial neural networks theory and its change-point detection algorithm. Chapter four describes the data used for the project. Chapter five focuses on results and discussion and chapter six is conclusions.

Chapter 2 Change-Point Detection: A Literature Review

Various authors have studied change-point detection problems using parametric and non-parametric procedures quite extensively. In some cases, the study was carried out for known underlying distributions, namely the binomial, Poisson, Gaussian and normal distributions, amongst others. This chapter discusses some of the work that has been done on change-point detection.

CUSUM is one of the widely used change-point detection algorithms. Basseville & Nikiforov (1993) described four different derivations for the CUSUM algorithm. The first is more intuition-based, and uses ideas connected to a simple integration of signals with an adaptive threshold. The second derivation is based on a repeated use of a sequential probability ratio test. The third derivation comes from the use of the off-line point of view for multiple hypotheses testing. The fourth derivation is based upon the concept of open ended tests.

The principle of CUSUM stems from stochastic hypothesis testing method (Chen, 1999). We define the following two hypotheses about the parameter θ :

$$\begin{cases} H_0: \theta = \theta_0 \\ H_1: \theta = \theta_1 \end{cases} \quad (3.1)$$

As long as the decision is taken in favour of H_0 there is no parameter change. Once the decision is taken in favour of H_1 , then there is a parameter change. The test for signalling a change is based on the log –likelihood ratio S_k :

$$S_k = \sum_{i=1}^k s_i = \sum_{i=1}^k \ln \frac{p_{\theta_1}(y_i)}{p_{\theta_0}(y_i)} \quad (3.2)$$

In this expression, k denotes the present time instant; y_i is the time series; θ_0 and θ_1 are the parameter values under the two hypotheses H_0 and H_1 , assuming that the variance σ^2 is the same, and s_i is a log-likelihood ratio of the time series.

The typical behaviour of the log-likelihood ratio s_k shows a negative drift before change and a positive drift after change. Therefore, the relevant information for

detecting a change lies in the difference between the value of the log-likelihood ratio and its current minimum value (Basseville & Nikiforov, 1993). Thus, the alarm condition for the CUSUM algorithm takes the following form:

$$g_k = S_k - m_k \geq h, \quad m_k = \min_{1 \leq j \leq k} S_j$$

where the alarm signal is at k , and the parameter h is a threshold parameter.

The change-point time is

$$t_a = \min\{k : g_k \geq h\} \quad (3.3)$$

The detection rule is a comparison between the cumulative sum S_k and an adaptive threshold g_k .

De Oca et al. (2010) used CUSUM for change-point detection in non-stationary sequences, applying it to data network surveillance. They proposed an algorithm that uses a defined time slot structure to take into account time varying distributions, and used historical samples of observations within each time slot to facilitate a nonparametric methodology. The algorithm includes an online screening feature that enables full automation of the algorithm and eliminates the need for manual oversight.

Lin et al. (2007) proposed an adaptive CUSUM algorithm (ACS) to detect an anomaly. The proposed algorithm reduces false alarm rates without lowering the detection probability. They incorporated a sliding model controller into a CUSUM detector. The adaptive CUSUM algorithm prevents unlimited build-up of the accumulator. That way ACS detects change-points at an onset and termination of an anomaly period, while satisfying the requirements of detection and false alarm time. Verdier et al. (2008) presented the optimality of CUSUM rule approximations in change-point detection problems and application to nonlinear state-space systems. Gavit et al. (2009) used CUSUM change-point algorithm in: 1) determining if process changes or improvements may have led to a shift in an output, 2) problem solving, and 3) trend analysis. They applied this tool in the pharmaceutical industry.

Nazario et al. (1997) developed a sequential test procedure for transient detections in a stochastic process that can be expressed as an autoregressive moving average

(ARMA) model. Preliminary analysis shows that if an ARMA(p, q) time series exhibits a transient behavior, then its residuals behave as an ARMA(Q, Q) process, where $Q \leq p + q$. They showed that residuals from the model before the parameter change behave approximately as a sequence of independent random variables - after a parameter change, the residuals become correlated. Based on this fact, they derived a new sequential test to determine when a transient behavior occurs in a given ARMA time series.

The drawback of the developed test was that it was capable of detecting only the parameter changes which cause significant alterations in the ACF of residuals. Thus, if a parameter change produces small alterations in the ACF of residuals, then these changes might not be detected.

Habibi et al. (2005) derived a test statistic T_n for change-point detection in a general class of distribution. The derived test statistics reduces to the statistic obtained by Kander & Zacks (1966) for the exponential family. Kawahara & Sugiyama, (2009) presented a novel non-parametric approach to detecting the change of probability distributions of sequence data. The key idea of the method is that the ratio of two probability densities is directly estimated without going through density estimation. The proposed method can avoid nonparametric density estimation, which is known to be difficult in practice.

The derived change-point detection algorithm is based on direct density-ratio estimation that can be computed very efficiently in an online manner. The drawback of the developed algorithm is that it requires the dimensionality of data samples to be high because not all observations at each time point are used; rather sequences of observations are treated as samples for estimation. This can potentially cause performance degradation in density-ratio estimation.

Staudacher et al. (2005) developed a method for change-point detection for online analysis of the heart beat variability during sleep. The method was developed from the detrended fluctuation analysis (DFA). The new method is called progressive detrended fluctuation analysis (PDFA). Although the method was developed as a tool for numerical sleep evaluation based on heart rate variability in the ECG-channel of polysomnographic whole recordings, it proves to be applicable to many time series systems. The method was successfully applied to numerous artificially

generated data sets of Gaussian random numbers, as well as to time series with non-stationarities, such as non-polynomial trends.

Blazek et al. (2001) developed efficient adaptive sequential and batch-sequential methods for an early detection of attacks from the class of “denial-of-service attacks”. Both the sequential and batch-sequential algorithms used thresholding of test statistics to achieve a fixed rate of false alarms. The algorithms are developed on the basis of the change-point detection theory: to detect a change in statistical models as soon as possible, controlling the rate of false alarms. There are three attractive features of the approach. First, both methods are self-learning, which enables them to adapt to various network loads and usage patterns. Second, they allow for detecting attacks with a small average delay for a given false alarm rate. Third, they are computationally simple, and hence, can be implemented online.

Lund et al. (2007) looked at the change-point detection in periodic and auto correlated time series using classic change-point tests based on sums of squared errors. The method developed by Lund et al. (2007) was successfully applied in the analyses of two climate changes.

Downey (2008) developed an algorithm for estimating the location of change-point in a time series that includes abrupt changes. The algorithm is Bayesian in the sense that the results are a joint posterior distribution of the parameters of the process, as opposed to a hypothesis test or an estimate value. The algorithm requires subjective prior probabilities, assuming that the change-points are equally likely to occur at any time, the required prior is a single parameter, f , the probability of a change-point. A drawback of the developed algorithm is that it requires time proportional to n^2 at each time step, where n is the number of steps from the second to last change-point. If the time between change-points is a few thousand, the computing costs become unrealistic.

Moskvina & Zhigljavsky (2003) developed an algorithm of change-point detection in time series, based on sequential application of the singular-spectrum analysis (SSA). The main idea of SSA is performing singular value decomposition (SVD) of the trajectory matrix obtained from the original time series with a subsequent reconstruction of the series.

This algorithm is based on the idea that if, at a certain time moment τ , the mechanism generating the time series x_t has changed, then an increase in the distance between a subspace of \mathbb{R}^M spanned by certain eigenvectors of the so-called lag-covariance matrix, and M -lagged vectors x_{j+1}, \dots, x_{j+M} is to be expected for $j \geq \tau$. They applied the algorithm to different data sets. The SSA algorithm was also compared with the CUSUM algorithm. They showed that for a mean change, the CUSUM algorithm was about three times better than the SSA-based algorithm; for a variance change the SSA-based algorithm was about six times better than the CUSUM algorithm.

Vaisman et al. (2010) applied singular spectrum-based change-point analysis (developed by Moskvina & Zhigljavsky, 2003) to EMG-onset detection. Its performance was found to be comparable to those that are currently used, i.e. Hodges and Bui's method and Donoho's wavelet-based denoising method. They concluded that the SSA algorithm has great potential for real-time applications involving prostheses and neuro-prostheses.

Mei, (2006) developed a general theory for change-point problems when both the pre-change distribution f_θ and the post distribution g_λ involve unknown parameters. The approach was applied in the case of detecting shifts in the mean of independent normal observations. Kawahara et al. (2007) proposed an algorithm for detecting change-points in time series data based on subspace identification. The proposed algorithm is derived from the principle that the subspace spanned by the columns of an observability matrix and the one spanned by the subsequences of time series data are approximately equivalent. The algorithm was compared with SSA and it promised to have a superior performance.

Mboup et al. (2008) presented a change-point detection method based on a direct online estimation of the signal's singularity points. Using a piecewise local polynomial representation of the signal, the problem is cast into a delay estimation. A change-point instant is characterized as a solution of a polynomial equation, the coefficients of which are composed by short time window iterated integrals of the noisy signal. The change-point detector showed good robustness to various types of noises.

Chen et al. (2007) developed a distributed change-point detection scheme (DCD) to detect flooding type DDoS attacks over multiple network domains. The DCD scheme detects DDoS flooding attacks by monitoring the propagation of abrupt traffic changes inside the network. If a CAT tree is constructed sufficiently large and the tree size exceeds a preset threshold, an attack is declared.

Vincent (1998) presented a new technique for the identification of inhomogeneities in Canadian temperature series. The technique is based on the application of four linear regression models in order to determine whether the tested series is homogeneous. Vincent's procedure is a type of "forward regression" algorithm in that the significance of the non-change-point parameters in the regression model is assessed before (and after) a possible change-point is introduced. In the end, the most parsimonious model is used to describe the data.

The chosen model is then used to generate residuals. It uses the autocorrelation in the residuals to determine whether there are inhomogeneities in the tested series. At first, it considers the entire period of time and then it systematically divides the series into homogeneous segments. Each segment is defined by some change-points, and each change-point corresponds to either an abrupt change in mean level or a change in the behavior of the trend.

Auret & Aldrich (2010) used random forest models to detect change-points in dynamic systems. Wei et al. (2010) used Lyapunov exponent and the change-point detection theory to judge whether anomalies have happened. Aldrich & Jemwa, (2007) used phase methods to detect change in complex process systems.

Shi et al. (2011) developed a novel technique to address multiple change-point detection problems using two key steps: 1) apply the recent advances in consistent variable selection methods such as SCAD, adaptive LASSO and MCP to detect change-points, and 2) employ a refined procedure to improve the accuracy of change-point estimation. They further established a connection between multiple change-point detection and variable selection through proper segmentation of data sequence.

The different methods and algorithms perform optimally in certain types of time series. There is no global technique applicable to all kinds of time series. The linear

change-point detection techniques works best in linear time series and tend to be inaccurate in nonlinear time series. Nonlinear change-point detection techniques work optimally for nonlinear time series. It is thus imperative to know the nature of the time series to be analyzed before choosing a technique that will be used for the analysis.

Chapter 3 Change-Point Detection with SSA and Artificial Neural Networks

This chapter covers the theory behind singular spectrum analysis and artificial neural networks. These two methods are subsequently used in developing a novel change-point detection technique in process monitoring.

3.1 Singular Spectrum Analysis (SSA)

The Singular Spectrum Analysis (SSA) technique is a novel and powerful technique of time series analysis incorporating elements of classical time series analysis, multivariate statistics, multivariate geometry, dynamical systems and signal processing (Hassani, 2007). SSA is applied to many fields, such as biodiversity management (Dana, 2010), monitoring of climate changes (Ghil et al., 2001), forecasting (Zhigljavsky et al., 2009; Beneki et al., 2009), tool wear detection (Salgado and Alonso, 2006), and machine fault detection (Kia et al., 2007; Wise & Gallagher, 1996). The aim of SSA is to make a decomposition of the original series into the sum of a small number of independent and interpretable components such as a slowly varying trend, oscillatory components and a structure less noise (Hassani, 2007). It consists of four stages, namely embedding, singular value decomposition, reconstruction and diagonal averaging. The algorithm for SSA is taken from Hassani (2007) and is discussed below.

Stage 1: Embedding

The first step in the SSA algorithm is the embedding step where the initial time series changes into a trajectory matrix, or a multi-dimensional series. Consider a time series

$$x_1, x_2, \dots, x_N \quad (3.1)$$

Let $M (M \leq N/2)$ be some integer called 'lag' and let $K = N - M + 1$

The time series (3.1) is mapped into a multi-dimensional time series called the trajectory matrix using an embedding window of length L where $L = M * K$.

Let $X = [X_1, X_2, X_3, \dots, X_K]$ denote the columns of the trajectory matrix, where

$$X_i = (x_1, x_2, \dots, y_{i+M-1})^T \in \mathcal{R}^M \quad (3.2)$$

For $i = 1, 2, \dots, k$

The trajectory matrix \mathbf{X} is a Hankel matrix, which means that all the elements along the diagonal $i + j = \text{constant}$ are equal. A trajectory matrix is shown in equation (3.3)

$$X_{i,j} = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_K \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & \dots & x_k \\ x_2 & x_3 & \dots & x_{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_M & x_{M+1} & \dots & x_N \end{pmatrix} \quad (3.3)$$

Stage 2: Singular Value Decomposition (SVD)

The second step, the SVD step, involves the singular value decomposition of the trajectory matrix, and represents it as a sum of rank-one bi-orthogonal elementary matrices. Denoted by $\lambda_1, \dots, \lambda_M$, the eigenvalues of XX^T in decreasing order of magnitude ($\lambda_1 \geq \dots \lambda_M \geq 0$), and by U_1, \dots, U_M , the orthonormal system of the eigenvectors of the matrix XX^T corresponding to these eigenvalues, (U_i, U_j) is the inner product of the vectors U_i and U_j and $\|U_i\|$ is the norm of vector U_i .

Set $d = \max(i, \text{such that } \lambda_i > 0) = \text{rank } \mathbf{X}$.

Denote $V_i = X^T U_i / \sqrt{\lambda_i}$, then the SVD of the trajectory matrix can be written as:

$$\begin{aligned} \mathbf{X} &= X_1 + X_2 + \dots + X_d \\ &= U_1 \sqrt{\lambda_1} V_1^T + \dots + U_d \sqrt{\lambda_d} V_d^T \\ &= \sum_{i=1}^d U_i \sqrt{\lambda_i} V_i^T \end{aligned} \quad (3.4)$$

where $X_i = U_i \sqrt{\lambda_i} V_i^T$ are bi-orthogonal matrices with rank-one, known as elementary matrices, thus rank of $\mathbf{X} = d$.

$$\|\mathbf{X}\|^2 = \sum_{i=1}^d \lambda_i \text{ and } \|X_i\|^2 = \lambda_i \text{ for } i = 1, \dots, d$$

The ratio of each eigenvalue (λ_i) to the sum total of all eigenvalues, that is, $\lambda_i / \sum_{i=1}^d \lambda_i$ can be calculated since $\|\mathbf{X}\|^2 = \sum_{i=1}^d \lambda_i$ and $\|X_i\|^2 = \lambda_i$ for $i = 1, \dots, d$. This ratio expresses the contribution of the matrix X_i to \mathbf{X} .

Similar to PCA, the spectral decomposition of the trajectory matrix $X \in \mathcal{R}^{M \times k}$ can be written as a product of a score matrix $T \in \mathcal{R}^{M \times k}$ and a transposed loading matrix. The trajectory matrix can be expressed as the sum of the outer product of the individual pairs of vectors t_i and p_i by setting $T_i = U_i$ and $P_i = V_i / \sqrt{\lambda_i}$

$$X = T_1 P_1^T + T_2 P_2^T + \dots + T_d P_d^T \quad (3.5)$$

Since SSA is simply PCA performed on the trajectory matrix, mathematical and statistical properties of PCA extend to SSA. The leading principal components capture most of the information when variables are highly correlated in the observation space. Data representation using the first few principal components (PCs) helps to separate signal and embedded noise in the data. Also, the first few PCs have minimal entropy with respect to the inputs (assuming data are normally distributed). The number of PCs that should be retained is application specific – a few PCs that explain between 60 and 80 % of variance might be considered adequate for visualization purposes but insufficient for modeling purposes. Figure 3.1 below shows a typical example of SSA-based time series decomposition.

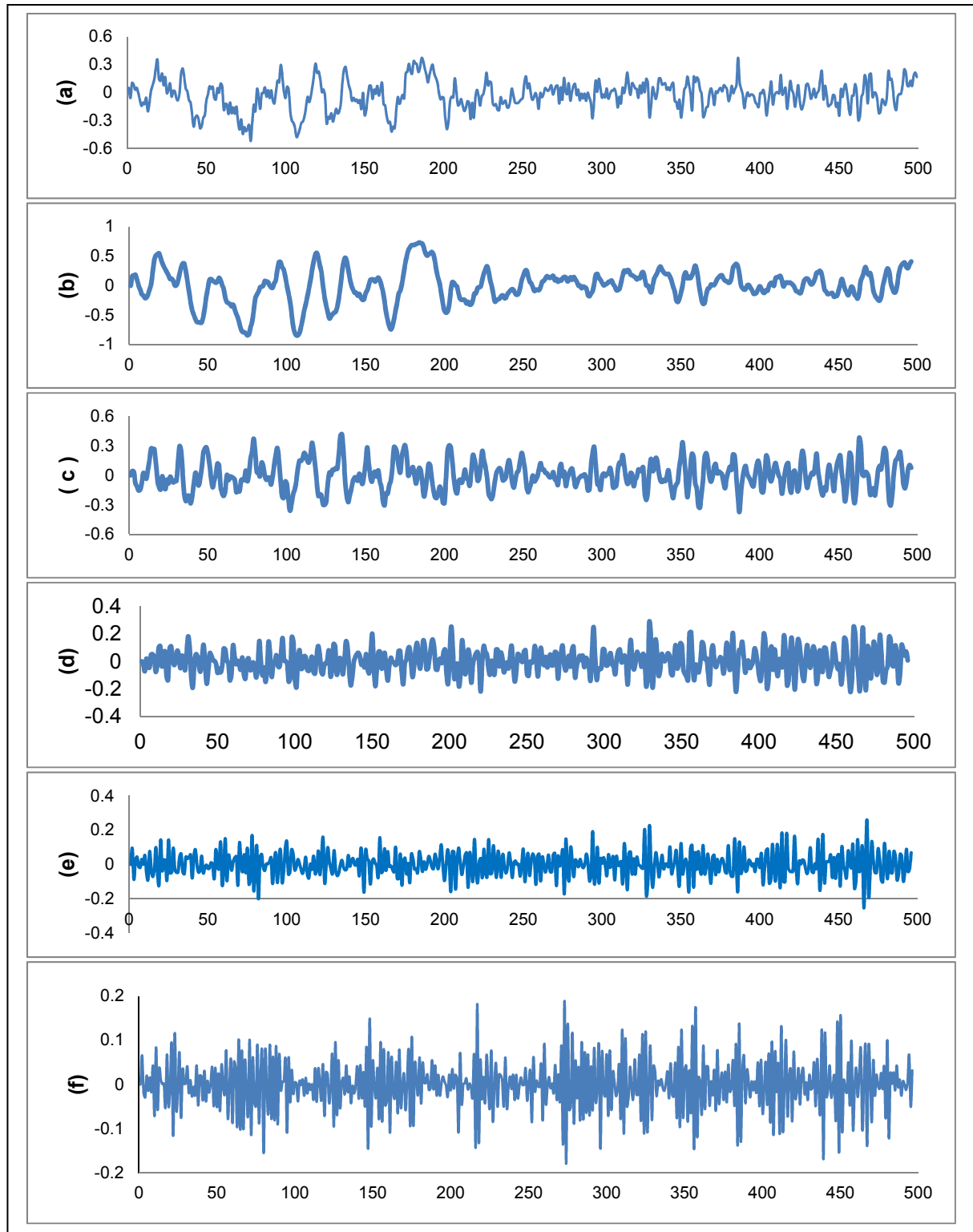


Figure 3.1: a) is the original time series, b) explains the most variation (68%), c) explains 17 %, d) explains 7 %, and e) and f) explain 5% and 3% respectively. The lower numbers are normally regarded as noise and are discarded. b) and c) combined explain 85 % of the variance which is sufficient and the other 15 % will be regarded as noise.

Stage 3: Reconstruction

The grouping step corresponds to splitting the elementary matrices X_i into several groups and summing the matrices within each group. The objective of this step is to separate the additive components of the time series by which the signal is expressed as the sum of intrinsic dynamical components and external noisy components.

Let $I = \{i_1, \dots, i_p\}$ be a group of indices. Then the matrix X_I corresponding to the group I is defined as:

$$X_I = X_{i1} + \dots + X_{ip}.$$

The split of the set of indices $J = 1, \dots, d$ into the disjoint subsets I_1, \dots, I_m corresponds with the representation

$$X = X_{I1} + \dots + X_{Im} \quad (3.6)$$

The procedure for choosing the sets I_1, \dots, I_m is called the eigentriple grouping. For a given group I the contribution of the component X_I into the expansion (3.6) is measured by the share of the corresponding eigenvalues: $\sum_{i \in I} \lambda_i / \sum_{i=1}^d \lambda_i$

Stage 4: Diagonal averaging

In the basic SSA algorithm, the diagonal averaging step is to transform the grouped matrices X_{Ii} into a new time series of length T . The time series \tilde{X}^i is obtained from averaging the corresponding diagonals of the matrix X_{Ii} .

$$\tilde{X}_{i,j} = \begin{cases} \frac{1}{s-1} \sum_{j=1}^{s-1} x_{j,s-j}, & \text{for } 2 \leq s \leq M \\ \frac{1}{M} \sum_{j=1}^N x_{j,s-j}, & \text{for } M+1 \leq s \leq K+1 \\ \frac{1}{N-s+2} \sum_{j=s-K}^{N-s+2} x_{j,s-j}, & \text{for } K+2 \leq s \leq N+1 \end{cases} \quad (3.7)$$

Let the Hankelization operator H be averaging the corresponding diagonals of the matrix X_{Ii} for $i = 1, \dots, m$. The Hankelization procedure uses the Hankelization operator H to transform X_{Ii} into:

$$\tilde{X}^i = HX_{I_i} \text{ for } i = 1, \dots, m \quad (3.8)$$

Under the assumption of weak separability, the initial time series is reconstructed by:

$$\mathbf{X} = \tilde{X}^1 + \tilde{X}^2 + \dots + \tilde{X}^m \quad (3.9)$$

Through diagonal averaging or Hankelization, the elementary matrix is transformed into a principal component of length N to create reconstructed components of the original series. The decomposition of the signal can be expressed as:

$$y_t = z_t + r_t, t = 1, 2, \dots, N \quad (3.10)$$

where y_t is the original time series, z_t is the reconstructed time series consisting of the chosen principal components and is associated with the deterministic components (trends), whilst r_t is associated with the stochastic components (noise) in the data. Singular spectrum analysis can be summarized by Figure 3.2:

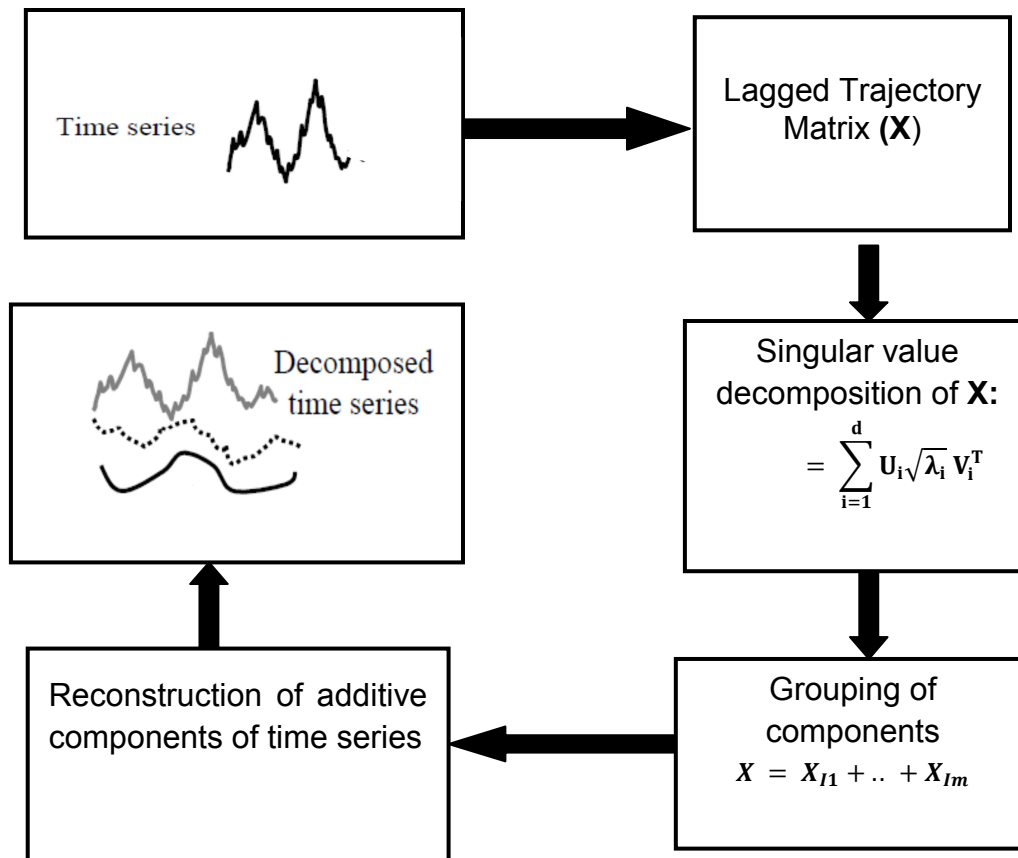


Figure 3.2: A schematic diagram of the SSA methodology.

3.2 Change-Point Detection with Singular Spectrum Analysis

The algorithm to detect the change-points in the data using SSA was developed by Moskvina and Zhigljavsky, (2003). The idea behind the algorithm is to apply SSA to a windowed portion of the signal. SSA picks up a structure of the windowed portion of the signal as an l -dimensional subspace.

If the signal structure does not change further along the signal, then the vectors of the trajectory matrix further along will stay close to this subspace. However, if the structure changes further along, it will not be well described by the computed subspace, and the distance of trajectory matrix vectors to it will increase. This increase will signal the change.

3.2.1 Algorithm

Let x_1, x_2, \dots, x_N be a time series, with N data points. Choose a window of width m and the lag parameter M , such that $M \leq \frac{m}{2}$. Set $K = m - M + 1$. Then, for each $n = 0, 1, \dots, N - m - M$, take an interval of time series $[n + 1, n + m]$ and define the trajectory matrix X^n , size M by K (Equation 3.11)

$$X_B^{(n)} = \begin{pmatrix} x_{n+1} & x_{n+2} & x_{n+3} & \cdots & x_{n+k} \\ x_{n+2} & x_{n+3} & x_{n+4} & \cdots & x_{n+k+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n+M} & x_{n+M+1} & x_{n+M+2} & \cdots & x_{n+m} \end{pmatrix} \quad 3.11$$

Equation 3.11 is called the base matrix in the change-point detection algorithm. The columns of the matrix $X_B^{(n)}$ are the vectors

$$X_j^{(n)} = (x_{n+j}, \dots, x_{n+j+M-1})^T \text{ with } j = 1, \dots, k. \quad 3.12$$

For each $n = 0, 1, \dots, N - m - M$

- 1) Compute the lag-covariance matrix
- 2) $R_n = X^{(n)} * (X^{(n)})^T$ 3.13
- 3) Determine M eigenvalues and the eigenvectors of R_n , and sort the eigenvalues in decreasing order.
- 4) Compute the sum of eigenvalues and the percentage of this sum that each eigenvalue contributes. The greater this percentage, the more important is the component corresponding to the eigenvalue.

- 5) Select the number of components to use for change-point detection. For change-point analysis, it was found that it works best to select a group of components that represent most of the signal. The number of components in this group is defined as L , and the choice of L remains fixed for all the X_n computed from the signal.
- 6) Chose two parameters of test interval p and q (both greater than 0) and define a test matrix T on an interval $[n+p+1, n+q+M-1]$

$$X_T^{(n)} = \begin{pmatrix} x_{n+p+1} & x_{n+p+2} & x_{n+p+3} & \cdots & x_{n+q} \\ x_{n+p+2} & x_{n+p+3} & x_{n+p+4} & \cdots & x_{n+q+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n+p+M} & x_{n+p+M+1} & x_{n+p+M+2} & \cdots & x_{n+q+M-1} \end{pmatrix} \quad 3.14$$

where $q = p + Q$.

The only requirement is that the interval defined by the choice of p and q allows forming a test matrix that includes at least one column of signal values different from the trajectory matrix columns.

- 7) Compute $D_n(T_n)$ statistic, the sum of squared Euclidean distances between the vectors of the test matrix T and L chosen eigenvectors of the lag-covariance matrix R_n (Equation 3.13)

$$D_n = \sum_{j=p+1}^q \left((X_j^{(n)})^T X_j^{(n)} - (X_j^{(n)})^T U U^T X_j^{(n)} \right) \quad 3.15$$

where $X_j^{(n)}$ are the vectors constituting the test matrix $X_T^{(n)}$, and U is a matrix consisting of L eigenvectors of R_n . The increase of the value of this statistic signals that the change has occurred.

The first way to estimate the change-point locations is to compute local minima of the $D_n(X_T^{(n)})$ function preceding its large values.

- 8) To find precise locations of change-points an additional CUSUM statistic calculation is needed. CUSUM statistics are computed for $n = 0 \dots N - m - M$: (Equation 3.16)

$$W_1 = S_1, \quad W_{n+1} = (W_n + S_{n+1} - S_n - \frac{k}{\sqrt{MQ}})^+, \quad n \geq 1, \quad 3.16$$

where $(a)^+ = \max\{0, a\}$ for any $a \in R$ and k is a small non-negative constant.

A reasonable value of k is $k = \frac{1}{(\sqrt[3]{MQ})}$, (Moskvina, 2001)

where $S_n = \frac{D_n}{v_n}$, v_n is an estimator of the normalized sum of squared distances D_n at time intervals, at which the hypothesis of no change can be accepted. v_n is effectively a variance of noise in the signal (Moskvina & Zhigljavsky, 2003).

The algorithm announces the structural change if for some n we observe $W_n > hI$

$$h = \frac{2t_\alpha}{MQ} \sqrt{\frac{1}{3} Q(3MQ - Q^3 + 1)}, \quad 3.17$$

where t_α is a $(1 - \alpha)$ - quantile of the standard normal distribution, then the change-point estimate is a first point with non-zero value of W_n before reaching this threshold (Moskvina, Valentina and Zhigljavsky, Anatoly 2003).

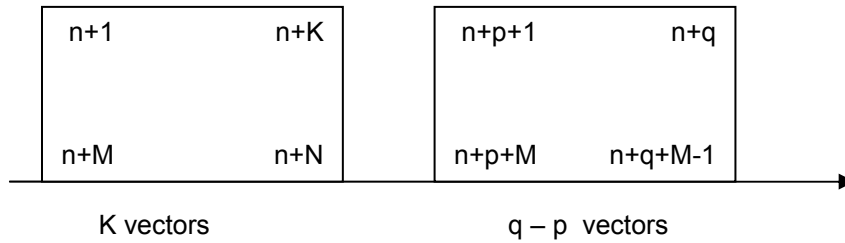


Figure 3.3: Construction of base and test matrices.

3.2.2 Choice of Parameters

Significant changes in time series structure will be detected for any reasonable choice of parameters (Moskvina & Zhigljavsky, 2003). To detect small changes in a noisy series, a tuning of parameters may be required.

- (i) Choice of lag M and number of features I : The recommendation is to choose $M = \frac{N}{2}$ and the first I components to provide a good description of the signal (Moskvina & Zhigljavsky, 2003).
- (ii) Length and location of the test sample: p, q : Choose $p \geq K$, in this case the columns of the base and the test matrices are generally different, and the algorithm is more sensitive to changes than when $p < K$. To get a smooth behaviour of the test statistics, q must be slightly larger than p .

- (iii) Window width N : The choice of N depends on the structural changes sought. If N is too large, changes can be missed or smoothed out. If N is too small, an outlier may register as a structural change

3.3 Neural Networks

An artificial neural network (ANN) is a system based on the operation of biological neural networks, i.e. an emulation of a biological neural system (Wythoff, 1993). It is amongst the most widely used nonlinear learning algorithms, inspired by Frank Rosenblatt's linear perceptron algorithm for classification (Rosenblatt, 1958).

An ANN is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well, neural networks learn by adjusting the weights of the connection lines.

Neural networks, with their ability to derive meaning from complicated or imprecise data, can be used to model almost every process (Jamali et al., 2007). ANNs have successfully been implemented to solve engineering and scientific problems in the areas of adaptive control, pattern recognition, machine vision, image processing, process diagnostics, process monitoring, and nonlinear system identification (Ramirez-Beltran & Jackson, 1999; Bishop, 1995; Bulsari, 1995; Dirion et al., 2002; Himmelblau, 2008). A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze (Virk et al., 2008). This expert can then be used to provide projections given new situations of interest. ANNs offer the prospect of a usable solution to problems that cannot even be described analytically (Wythoff, 1993).

Neural networks can be divided into two groups, depending on the basis functions used (Bulsari, 1995).

1. The first group is based on global basis functions. The well-known Multilayer Perceptron (MLP) belongs to this group.

2. The second group is called the localized receptive functions. The radial basis function networks (RBFN), the cerebellar model articulation controller (CMAC) and fuzzy networks belong to this second class of networks. Their input space is spanned by localized receptive functions.

In this work, only the multilayer perceptron will be discussed.

3.3.1 Multilayer Perceptron (MLP)

The MLP comprises a great number of highly interconnected elementary nonlinear neurons (or nodes), which are the processing elements where computations are carried out. The nodes are divided into disjoint subsets, called layers. Each node forms a weighted sum of the inputs from previous layers to which it is connected, adds a threshold (bias) value and produces its signal a , called the activation,

$$a = \sum_{j=1}^n w_{ij}x_j + \theta_i \quad (3.18)$$

Where $x_j (j = 1, \dots, n)$, $w_{ij} (j = 1, \dots, n)$ and θ_i are the inputs, weights and threshold (bias) associated to the unit i , respectively.

The activation signal, a , is sent to a transfer function, g . The transfer function can be any mathematical function, but is usually taken to be a simple bounded differentiable function such as the sigmoid or the arctangent function,

$$g(x) = \frac{1}{\pi} \tan^{-1}(x) + \frac{1}{2} \quad (3.19)$$

$$\text{or } g(x) = \frac{1 - e^x}{1 + e^x} \text{ or } g(a) = \frac{e^x}{1 + e^x} \quad (3.20)$$

Equation (3.20) is the sigmoid function.

The output of each node is therefore:

$$\text{out}_1 = g\left(\sum_{j=1}^n w_{ij}x_j + \theta_i\right) \quad (3.21)$$

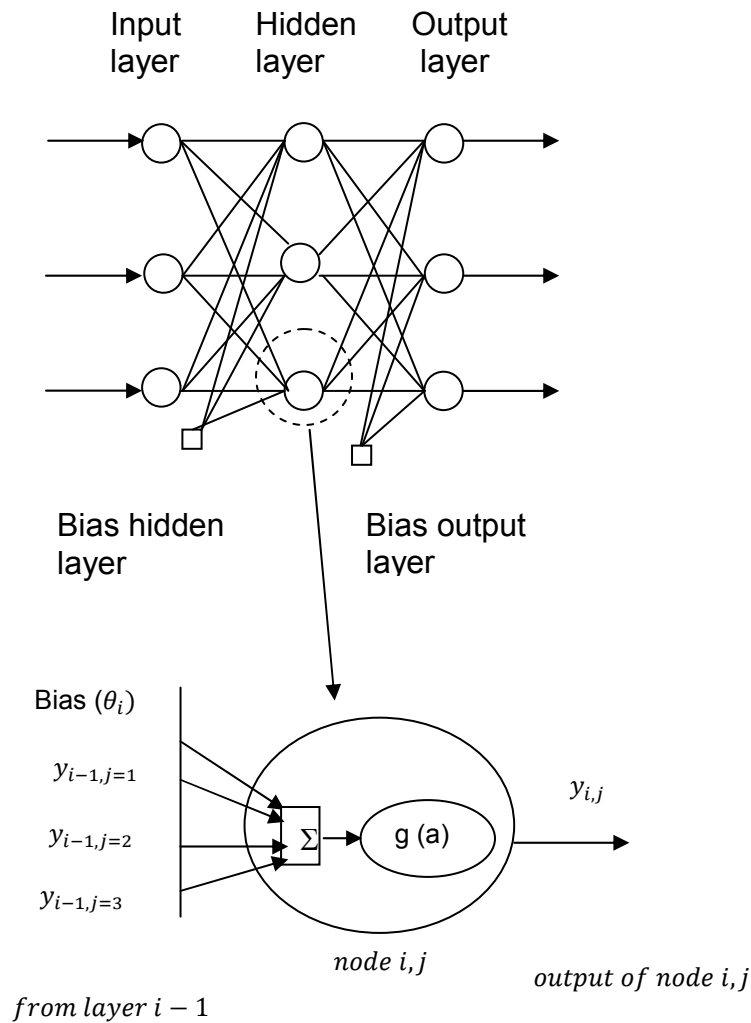


Figure 3.4: Schematic diagram of neural network.

This output value serves as an input to the next layer to which the node is connected, and the process repeated until output values are obtained in the output layer. The overall function of such a neural network is to compute an output vector from an input vector. The cells in the hidden layers are employed to increase the number of parameters and the nonlinear character of the neural network.

In order for the network to determine the function that maps the input vector to the output vector, it first undergoes training. The training step is called the “learning phase.” The data is divided into training data and test data. Once the network has been trained it is then fed with the test data to determine whether the output corresponds to the measured data. The training and test data should have an input vector and the target vector. The target vector is the solution. The learning concept is

based on determination of the weights and bias of the network. Learning is achieved by minimising a cost function.

$$E = \frac{1}{2} \sum_{p=1}^P \sum_{i=1}^{N_0} (O_i - T_i)^2 \quad (3.22)$$

Where N_0 and P represent the number of output neurons and the number of examples of the learning database, and O_i and T_i are the i^{th} output and the i^{th} target element, respectively.

The learning phase can be viewed as a parametric identification method to optimize the weights in the neural model. There are a couple of learning algorithms available for training the networks. Some of the popular learning algorithms are gradient descent, conjugate gradients, quasi-Newton methods and the Levenberg-Marquardt algorithm (Bishop, 1995). Different algorithms perform best on different problems and therefore it is difficult to single one out as the best. The most widely used is the back propagation algorithm (Rumelhart et al, 1986; Dirion et al., 2002), which is based on the steepest descent method - a derivative of the gradient descent algorithm.

However, the major disadvantages of back propagation algorithm are that its convergence rate is relatively slow, and being trapped at the local minima (Eckmiller & Malsburg, 1988).

3.3.2 Data Pre-processing

A neural network in principle can be used to map the raw input data directly onto the required final output values. Such an approach gives poor results, in most cases. For most applications, it is necessary to begin by transforming the data into some kind of representation before training a neural network. In some applications, the choice of pre-processing will be one of the most important factors in determining the performance of the final system. Process data often suffers from a number of deficiencies such as missing input values or incorrect target values. Data pre-processing or data preparation plays an important role in minimizing the impact of those deficiencies. This reduces the dimensionality of the data and may allow learning algorithms to operate faster and more effectively.

3.3.2.1 Input Normalization and Encoding

One of the most common forms of preprocessing consists of a simple linear rescaling of the input variables. This is useful when different variables have typical values that differ significantly. In a system monitoring a chemical plant for instance, two of the inputs might represent temperature and pressure respectively. They may have values that differ by several orders of magnitude. The typical sizes of the inputs may also not reflect their relative importance in determining the required outputs (Kotsiantis et al., 2006).

Linear transformation restructures all the inputs to have similar values. This is achieved by treating input variables independently, for each variable x_i we calculate its mean \bar{x}_i and variance σ_i^2 with respect to the training set, using

$$\bar{x}_i = \frac{1}{N} \sum_{n=1}^N x_i^n \quad (3.23)$$

$$\sigma_i^2 = \frac{1}{N} \sum_{n=1}^N (x_i^n - \bar{x}_i)^2 \quad (3.24)$$

where $n = 1, \dots, N$ labels the patterns. We then define a set of re-scaled variables given by:

$$\tilde{x}_i^n = \frac{x_i^n - \bar{x}_i}{\sigma_i} \quad (3.25)$$

The transformed variables have zero mean and unit standard deviation.

3.3.3 Limitations of Neural Networks

Although neural networks can be used to model almost every process, they have shortcomings that make their application unreliable. They are prone to over-fitting and under-fitting, and they can also be trapped on local minima.

3.3.3.1 Over-fitting and Under-fitting

The point of training an ANN on a training set of data is to represent subsequently process data not in the training set. The ANN should be able to generalize for other similar data. Neural networks are very flexible; their flexibility sometimes leads to either over-fitting or under-fitting depending on the degree of smoothing. With

enough hidden layers, the network can fit the data, including the noise, to arbitrary accuracy. Thus for a network with many parameters, reaching the global minimum may mean nothing more than finding a badly over-fitted solution. A network with fewer parameters may fail to detect fully the signal in complicated data, leading to under-fitting. Figure 3.5 below represents both over-fitting and under-fitting depending on how the data should be modelled. If the data is supposed to be modelled by a smooth curve, Figure 3.5(a) is over-fitting, whereas if it is a spectroscopic data with lots of detail, then Figure 3.5(b) is under-fitting.

Many approaches have been proposed to alleviate this problem (Bishop, 1995; Himmelblau, 2008; Hsieh, 2004; Coulibaly et al., 2000):

1. Use lots of data relative to the number of coefficients in the ANN.
2. Select a satisfactory model that contains the fewest weights possible.
3. When training, do not train excessively. As the number of iterations of the optimization steps increase, the error in the predictions by the training set will continue to decrease ever more slowly as the fit of the ANN increases. Stopping timeously will minimise the chances of over-fitting.
4. Train for some number of iterations, then stop, and test the model on the test data.

Another method used to minimize chances of over-fitting is to add weight penalty terms to the cost function.

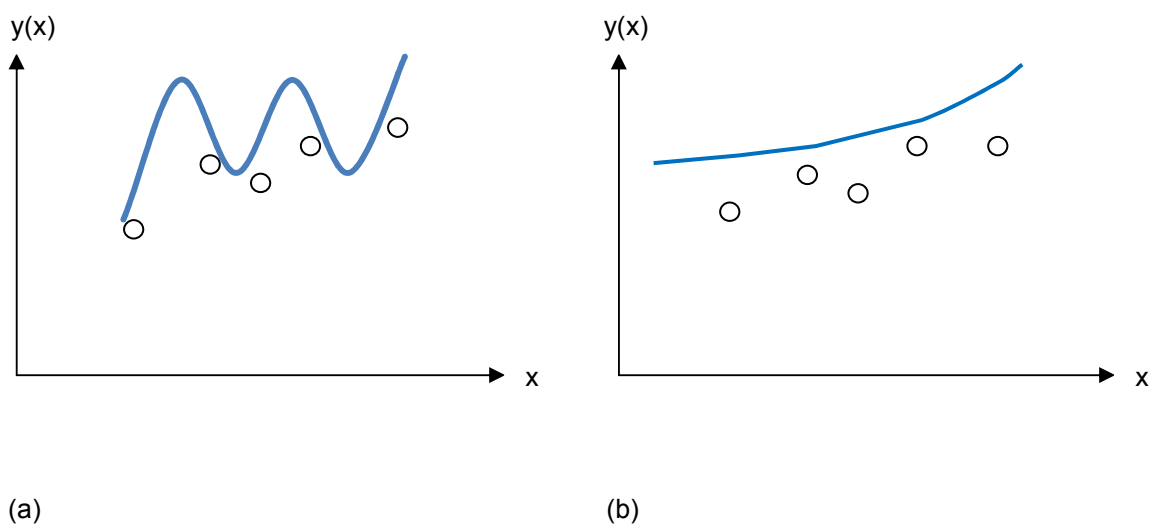


Figure 3.5: Two different functions used to fit the same data taken from (Himmelblau, 2008).

3.3.3.2 Local Minima

The main difficulty of the neural network (NN) is that the nonlinear optimization often encounters multiple local minima in the cost function. Figure 3.6 below is a schematic diagram showing local minima encountered in multilayer perceptron training using gradient descent search. The two local minima sites, as indicated by the arrows, risk the MLP failing to reach the global minimum, resulting in poor performance of the trained model.

This means that starting from different initial guesses for the parameters, the optimization algorithm may converge to different local minima. A number of methods and approaches have been proposed to minimize the chances of the network being trapped in local minima (Hsieh, 2004; Wythoff, 1993; Hsieh & Tang, 1998). They include training the network starting from different random initial parameters with the hope that not all the runs will be trapped to local minima. Hsieh and Tang, (1998) suggested the use of an ensemble of NNs, where their parameters are randomly initialized before training. The individual NN solutions will be scattered around the global minimum, but by averaging the solutions from the ensemble members, a better estimate of the true solution is obtained.

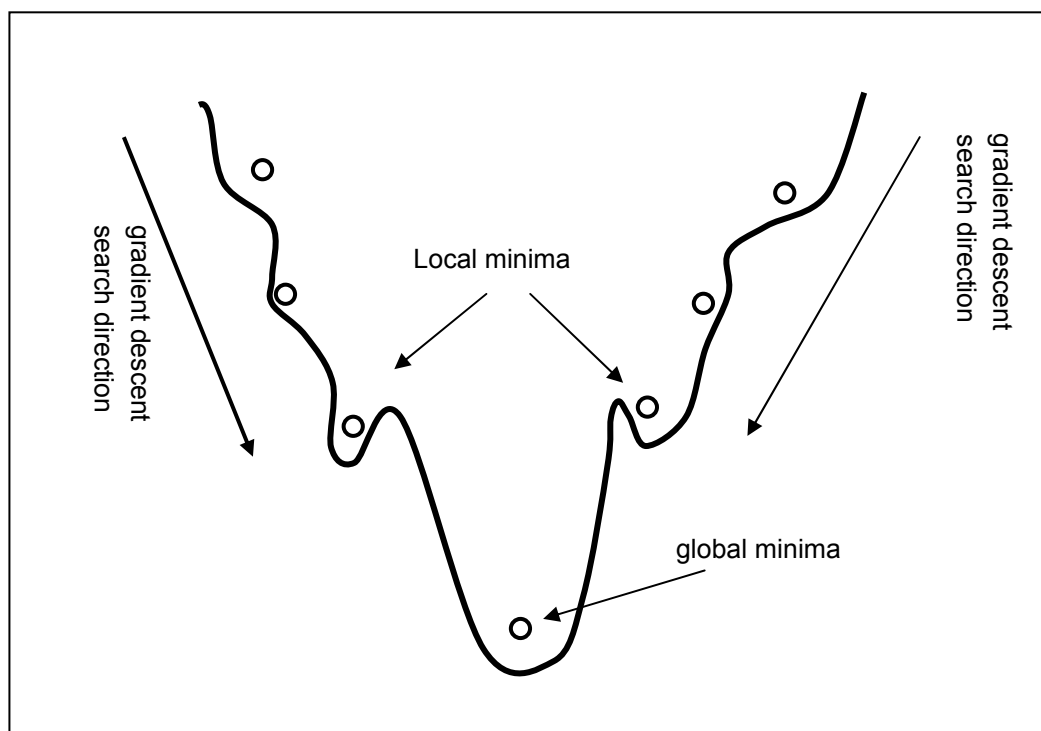


Figure 3.6: Local minima traps encountered in multilayer perceptron training using gradient descent algorithms (Jemwa, 2003).

An alternative ensemble technique is “bagging” (abbreviated from bootstrap aggregating) (Breiman, 1996). First, time series data is divided into a training set and a test set. The training set is also divided into equal subsets. An ensemble of NNs models are trained using a subset randomly drawn from the training set. The subset is drawn at random with replacement from the training set. The test set is used for testing the models, and the model that best fits the test set is chosen.

3.3.4 Auto-associative Neural Networks (NLSSA/NLPCA)

Neural networks have many structures depending on the intended use. The structure of particular interest in this work is the auto-associative neural network. The defining feature of this type of structure is the bottleneck. The middle hidden layer has fewer nodes - only one or two. Figure 3.7 below shows a typical structure of an auto-associative neural network.

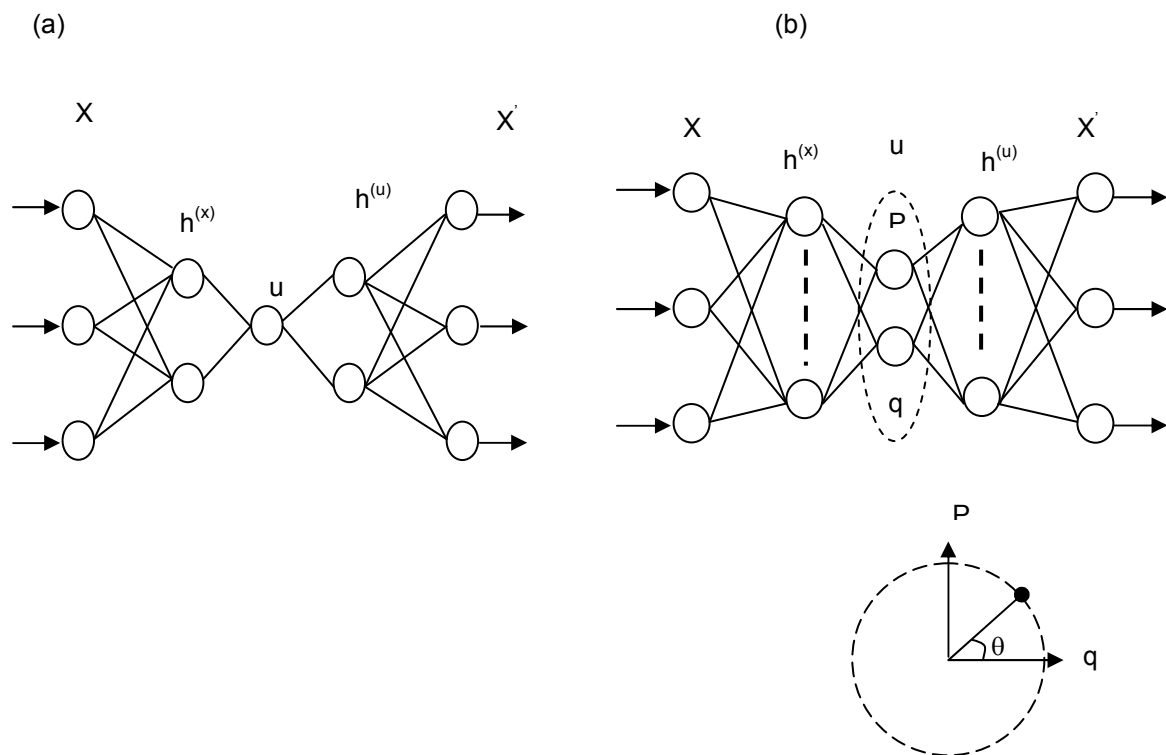


Figure 3.7: Schematic diagram of an auto-associative neural network (Hsieh, 2004).

Auto-associative neural networks are sometimes referred to as nonlinear principal component analysis (NLPCA) or nonlinear singular spectrum analysis (NLSSA) depending on its application (Hsieh, 2004).

Figure 3.7 a) is a schematic diagram of the auto-associative neural network model for calculating the nonlinear singular spectrum analysis (NLPCA). There are three

layers of hidden neurons located between the input layer x on the left and the output layer x' on the right. The hidden layer on the left ($h^{(x)}$) is the encoding layer, followed by the “bottleneck layer” (u) (with a single neuron). The hidden layer on the first right of the bottleneck layer ($h^{(u)}$) is the decoding layer.

Figure 3.7 b) is a schematic diagram of an auto-associative neural network with a circular node at the bottleneck (NLPCA(cir)). Instead of having one bottleneck neuron u , there are now two neurons p and q constrained to lie on a unit circle in the p - q plane, so there is only one free angular variable θ , the NLPC (Hsieh, 2004).

Nonlinear principal component analysis technique is mainly used in multivariate data analysis, similar to the well-known method of principal component analysis. NLPCA, like PCA, is used to identify and remove correlations amongst problem variables as an aid to dimensionality reduction, visualization, and exploratory data analysis. While PCA identifies only linear correlations between variables, NLPCA uncovers both linear and nonlinear correlations, without restriction on the character of the nonlinearities present in the data. NLPCA operates by training a feed forward neural network to perform the identity mapping, where the network inputs are reproduced at the output layer (Kramer, 1991). The bottleneck layer forces the network to develop a compact representation of the input data.

The similarities between PCA and SSA also exist between NLPCA and NLSSA. Nonlinear principal component analysis technique is mainly used in multivariate data analysis whilst NLSSA is mainly used in time series analysis. Given a time series $x_i = x_j$ ($j = 1, \dots, N$), embedding the time series yields the augmented matrix X ,

$$X = \begin{pmatrix} x_1 & x_2 & x_3 & \cdots & x_{N-L+1} \\ x_2 & x_3 & x_4 & \cdots & x_{N-L+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_L & x_{L+1} & x_{L+2} & \cdots & x_N \end{pmatrix} \quad (3.26)$$

The augmented matrix is similar to multivariate data. The procedure followed in NLPCA is applied to the augmented matrix.

The nonlinear principal component analysis (NLPCA) is a general purpose feature extraction algorithm producing features that retain the maximum possible amount of information from the original data set, for a given degree of data compression.

Information preservation assures that the selected features will be useful in most situations, independent of the ultimate application (Kramer , 1991).

A nonlinear function maps from the higher-dimension input space to the one-dimensional bottleneck space, followed by an inverse transform mapping from the bottleneck space back to the original space represented by the outputs, $\mathcal{H}^L \rightarrow \mathcal{H}^P \rightarrow \mathcal{H}^L$, where L is the dimension of the input and output data. P is the dimension of the data in the bottleneck which is equal to the number of nodes of the bottleneck.

Consider a data set $X(t_n) \in \mathcal{H}^L, n = 1, \dots, N$, define a function $h: \mathcal{H}^L \rightarrow \mathcal{H}^P$ and $g: \mathcal{H}^P \rightarrow \mathcal{H}^L$, $1 \leq P < L$ we optimize h and g such that $\epsilon = \sum_{i=1}^N (x_i - x'_i)^2$ is minimized, where, $x'_i = g \circ h(x_i)$

3.3.5 Feature Extraction

Feature extraction refers to identifying the salient aspects or properties of data to facilitate its use in a subsequent task, such as regression or classification. It is the process of efficiently capturing information contained in data. At the simplest level, feature extraction involves discarding a subset of the original data. A more complex approach implies the computation of linear or nonlinear combinations of all the original variables.

A large variety of feature extraction methods appears in the literature, based on statistical pattern recognition or on artificial neural networks (Lerner et al, 1996). In all the methods, a mapping G transforms a pattern y of an m-dimensional feature space to a pattern T of an f-dimensional projected space, $f < m$, i.e.:

$$T = G(y) \quad (3.27)$$

such that a criterion J is optimized. The mapping $G(y)$ is determined amongst all the transformations $g(y)$, as the one that satisfies:

$$J\{G(y)\} = \max J\{g(y)\}. \quad (3.28)$$

The number of neurons in the bottleneck determines the order of reduction. There are two ways of extracting features in an auto-associative neural network (Kerschen & Golival, 2003):

1. A non-modal analysis: a single neural network with f nodes in the bottleneck layer is used. This type of configuration is considered optimal in that it results in a minimal MSE and p -dimensional surface is learnt from the data. This network will result in f -principal components (Figure 3.8 below).
2. A modal analysis: f neural networks with a single node in the bottleneck layer are considered. The residual of the i^{th} network is the input of the $(i + 1)^{\text{th}}$ network. From a geometrical point of view, a series of f curves i.e. one dimensional approximation, is learnt (Figure 3.9 below).

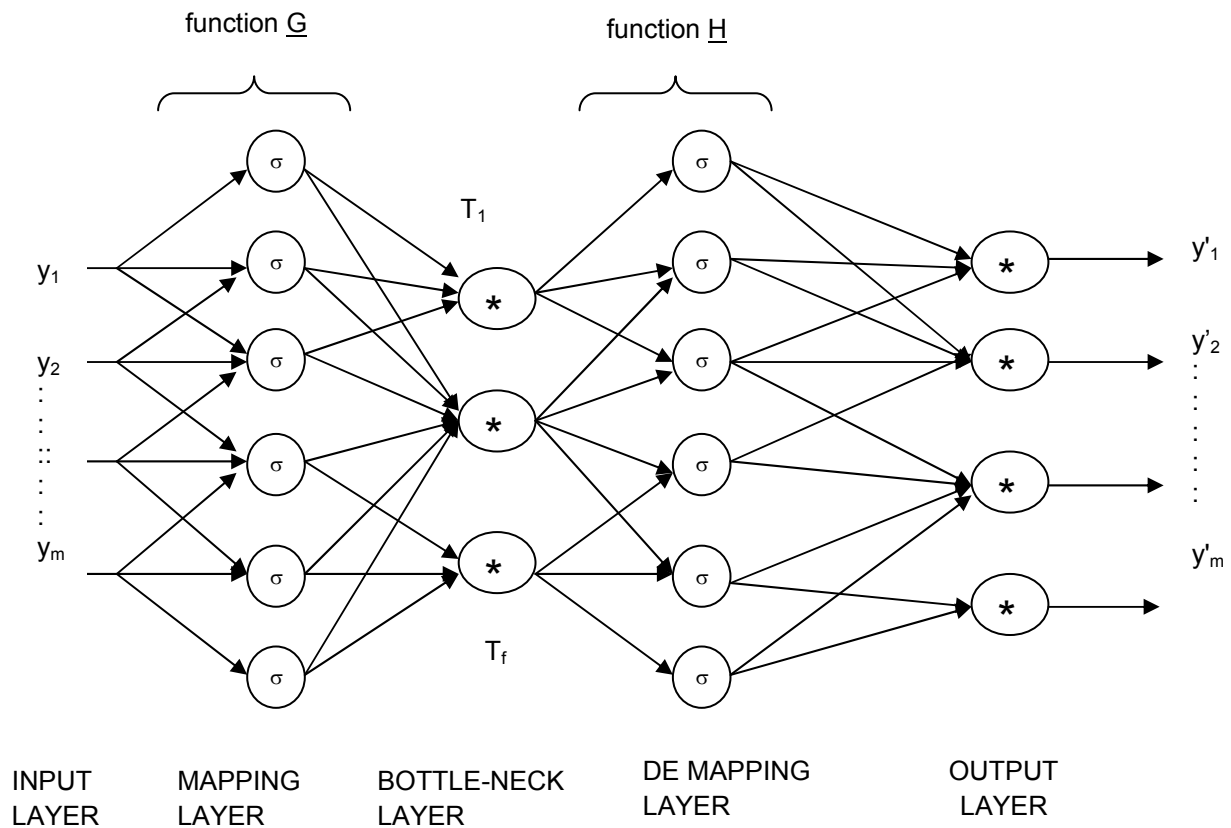


Figure 3.8: Network architecture for simultaneous determination of f nonlinear factors using an auto-associative network, taken from (Kramer, 1991). (σ Indicates sigmoidal nodes, $*$ indicates linear nodes).

The use of features is motivated by several factors (Kerschen & Golnival, 2003):

- It is difficult to extract valuable information from data just from the visual inspection of its time response. The use of features should help to alleviate this problem.

- Multi-variate data occupying n dimensions may be highly redundant and are almost never n -dimensional. The elimination of a significant number of dimensions to obtain an efficient representation of the underlying process should facilitate the analysis of the process behaviour. The fact that such dimensionality reduction can lead to better performance may appear at first as a paradox, since in most cases it results in loss of information. Dimension reduction techniques do not create new information, rather they allow relevant information contained in the signal to be enhanced.
- Noise always corrupts the data, and features may be less sensitive to it.

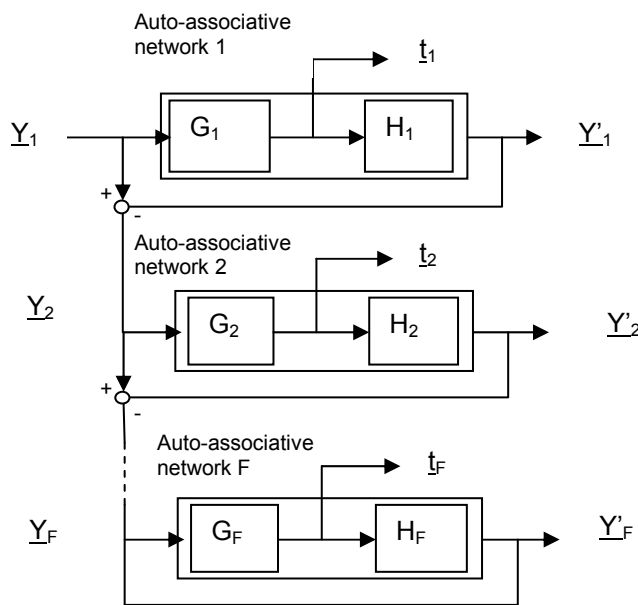


Figure 3.9: Sequential determination of a nonlinear factor by training F networks with one bottleneck node each, taken from (Kramer, 1991).

3.4 Nonlinear SSA with Auto-Associative Neural Networks

Nonlinear feature extraction using singular spectrum analysis (NLSSA) can be achieved by replacing PCA component with an auto-associative neural network. The network has three hidden layers. It has a bottleneck that is the middle layer which has fewer nodes compared to the other two hidden layers. The bottleneck is used in the extraction of features. It reduces the dimension of the state space from n to m , where $m < n$. Then again, from the bottleneck to the output layer, the dimension of the state space is increased from m to n , where $m < n$. The number of nodes in the input layer is equal to the dimension of the input matrix, i.e. embedding dimension. The output layer has the same number of nodes as the input layer since the input

and the target matrices are the same. The number of nodes for the first and third hidden layers depends on the complexity of the data to be modeled.

In an auto-associative network, the number of nodes for the first and third hidden layers is equal in number. The network build is normally started with fewer nodes in hidden layers one and three, the Schwartz Information Criteria (SIC) calculated for each run for a couple of runs, and then the number of nodes increased by one for each of the two hidden layers (i.e. a couple of networks are trained and the best one picked as the model). The procedure is repeated until a network that models and generalizes the data well and minimize the SIC is found.

The selection of the number of nodes in the bottleneck layer determines the order of reduction. The fraction of explained variance (FEV) can be calculated (Kerschen & Golival, 2003) by:

$$FEV = \frac{E[\|\hat{x}(t)\|^2]}{E[\|x(t)\|^2]} \quad (3.29)$$

The FEV indicator is similar to the eigenvalues of the covariance matrix that explain the energy percentage captured by the PCA (Kerschen & Golival, 2003). To attain a prescribed FEV, the number of nodes on the bottleneck layer is gradually increased until the prescribed FEV is achieved (Kerschen & Golival, 2004).

A trained auto-associative neural network learns the underlying structure of the system producing the data. The bottleneck layer forces the network to learn the important features of the system, eliminating redundancies and insignificant aspects of the data. Once the network learns to map the input data onto itself through the bottleneck, it should be able to map new data that was not used for training onto itself, as long as it is from the same system as the training data. The only time it fails to map the data onto itself is when there has been a parameter change in the system producing the data. Although the network is able to map the data onto itself, it still produces residuals, although these are normally small. Once there has been a parameter change in the process that produces the data, the magnitude of the residuals increase. Change-point detection using auto-associative neural networks use the principle of change in the magnitude of the residuals to detect change-points in systems.

3.4.1 Methodology for Change-Point Detection Using NLSSA

The methodology consists of five steps.

1. Data preparation – calculating time delay, embedding dimension and finally embedding of the time series (phase space reconstruction).
2. Network training - model calibration using a section of the ‘in control’ part of the time series.
3. Residual generation using the part of the time series that was not used for network training.
4. Determining the threshold value – this is the maximum value of the residuals, calculated using the ‘in control’ part of the time series. Any value above it is considered out of process limits.
5. Calculating change-point detection statistics.

These steps are explained in detail in the following paragraphs.

3.4.1.1 Time Series Embedding

Determination of embedding parameters namely time delay (τ) and embedding dimension (d) is the most important step in the nonlinear time series modeling. Various techniques have been developed for estimating embedding parameters, d and τ . Different techniques works best for different types of time series. In this work the most commonly used will be used.

- (i) *Time delay*: The two most commonly used methods for calculating time delay are linear autocorrelation function (ACF) and Average mutual information. Both methods are discussed in appendix (B1 for ACF and B2 for AMI).
- (ii) *Embedding Dimension*: False nearest neighbour (FNN) and Correlation dimension (CD) are the two most used methods for the calculation of embedding dimension. (See appendix for details of both methods, C1 for FNN and C2 for CD.)

3.4.1.2 Network Training Parameters

The neural network function¹ `newff`, uses 60 % of the training data to train the network, 20% validate the network and the remaining 20% to test the trained network. The network training stops when the cost function target has been achieved, the network errors start to increase instead of decrease, or it has reached the maximum number of iterations. During network training, the number of nodes for the second and fourth layers is increased gradually until the target cost function value has been achieved. Because of the tendency of neural network to be trapped in local minima, several networks are trained and the one with the lowest mean square error and the one that generalizes the function well is chosen as the correct model.

The activation function '`tansig`' is used for the first and third hidden layers and '`purelin`' is used for the second hidden layer (bottleneck) and output layer. The network is trained using the '`trainlm`' function. `Trainlm` updates weight and bias values according to Levenberg-Marquardt optimization. Performance function '`msereg`' is used. It measures network performance as the weight sum of two factors, the mean squared error and the mean squared weight and bias values.

$$msereg = \gamma * mse + (1 - \gamma)msw \quad (3.30)$$

where γ is the performance ratio, mse is the mean square error and msw is the mean square weight.

$$mse = \frac{1}{N} \sum_{i=1}^N (t_i - a_i)^2 \quad (3.31)$$

where t_i is the observed value and a_i is the simulated value.

$$msw = \frac{1}{n} \sum_{j=1}^n (w_j)^2 \quad (3.32)$$

¹ All referenced functions in Courier font are from the neural networks toolbox of MATLAB®

The performance ratio γ is varied between the range 0.5 -1, depending on the nature of the time series.

3.4.1.3 Model Calibration

Mathematically the set of MLP model structures can be defined as:

$$M_{FF}^* = \{M_{FF}(\theta) | \theta \in D_M \subset \mathcal{H}^d\} \quad (3.33)$$

where θ is the parameter vector of a model, d the order of the model, and D_M the set of possible model parameters. M_{FF} Denotes a model structure while $M_{FF}(\theta)$ indicates a specific model for the estimated parameter vector θ . In this work, a trained neural network model will be denoted by $M_{FF}(\theta)$.

Given a time series $x(i) = x_1, x_2, \dots, x_p, \dots, x_N$,

where x_1 to x_p , is part of the time series before the parameter change or 'in control region'. N is the length of the time series.

- Choose $M < \frac{p}{2}$, then for each $n = 1, 2, 3, \dots, M$, make an interval time series.

$$x(i)_t = x_1, x_2, x_3, \dots, x_M \quad (3.34)$$

Use AMI (equation B1, in the Appendix) to calculate the embedding lag (τ) and FNN (equation B7 in the Appendix) to calculate the embedding dimension (d). Using (τ) and (d) create a lagged matrix.

$$X_t = \begin{pmatrix} x_n & x_{n+1} & x_{n+2} & \cdots & x_{M-(d-1)\tau} \\ x_{n+\tau} & x_{n+1+\tau} & x_{n+2+\tau} & \cdots & x_{M-(d-2)\tau} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n+(d-1)\tau} & x_{n+(d-1)\tau+1} & x_{n+(d-1)\tau+2} & \cdots & x_M \end{pmatrix} \quad (3.35)$$

- Use X_t to train the neural network and obtain the model $M_{FF}(\theta)$

3.4.1.4 Residual Generation

- Choose L such that $M < L < p$, then for each $n = M + 1, M + 2, \dots, L$, make an interval time series.

$$x(i)_{threshold} = x_{M+1}, x_{M+2}, x_{M+3}, \dots, x_{L1} \quad (3.35)$$

- Using (τ) and (d) create a lagged matrix:

$$X(i)_{threshold} = \begin{pmatrix} x_{M+1} & x_{M+2} & x_{M+3} & \cdots & x_{L-(d-1)\tau} \\ x_{M+1+\tau} & x_{M+2+\tau} & x_{M+3+\tau} & \cdots & x_{L-(d-2)\tau} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{M+1+(d-1)\tau} & x_{M+(d-1)\tau+2} & x_{M+(d-1)\tau+3} & \cdots & x_L \end{pmatrix} \quad (3.36)$$

- Use $X(i)_{threshold}$ to simulate the output:

$$Y(i)_{threshold} = \text{sim}(M_{FF}(\theta), X(i)_{threshold}) \quad (3.37)$$

$$Y(i)_{threshold} = \begin{pmatrix} y_{M+1} & y_{M+2} & y_{M+3} & \cdots & y_{L-(d-1)\tau} \\ y_{M+1+\tau} & y_{M+2+\tau} & y_{M+3+\tau} & \cdots & y_{L-(d-2)\tau} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{M+1+(d-1)\tau} & y_{M+(d-1)\tau+2} & y_{M+(d-1)\tau+3} & \cdots & y_L \end{pmatrix} \quad (3.38)$$

- Unembed $Y(i)_{threshold}$ and create a time series,

$$y(i)_{threshold} = y_{M+1}, y_{M+2}, y_{M+3}, \dots, y_{L1} \quad (3.39)$$

- Calculate residuals:

$$r(i)_{threshold} = (x(i)_{threshold} - y(i)_{threshold})$$

Figure 3.10 shows a schematic summary of change-point detection using residual generation.

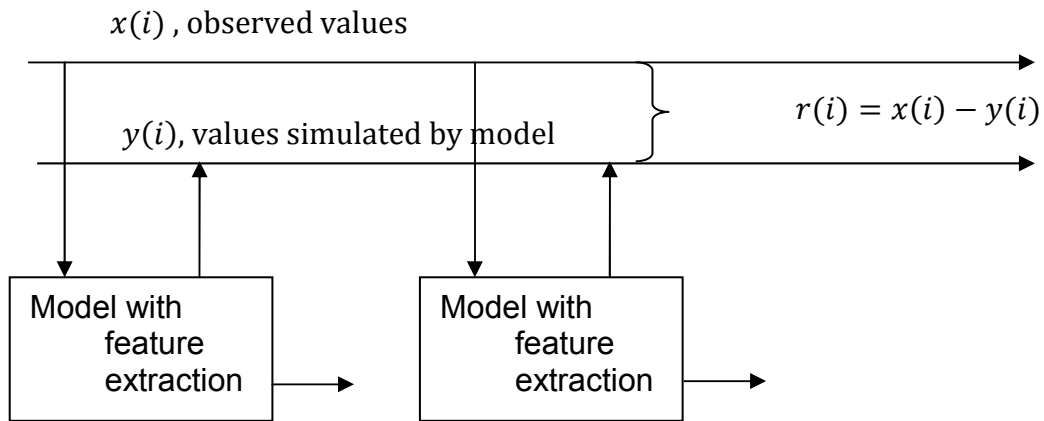


Figure 3.10: Schematic diagram showing change-point detection using residual generation.

3.4.1.5 Determination of the Threshold Value and Change-Point Statistics

Subsequent to residual generation is the evaluation of residuals to decide the likelihood of whether a parameter change has occurred. Because of unavoidable

modeling uncertainties, as well as system measurement noise, robust evaluation is required to avoid or minimize false alarms and failure to detect process deviations from normal conditions. The residual evaluation procedure involves choice of the evaluation function and corresponding detection thresholds. During process operation, the output from the preceding residual generation step is input into a residual evaluation function. The resulting output is compared to the threshold limits and, depending on whether or not a violation of the threshold has occurred, an appropriate decision is made.

$$r(i)_{threshold}^2 = (x(i)_{threshold} - y(i)_{threshold})^2 \quad (3.40)$$

Calculate 95% or 99% quantile $r(i)_{threshold}^2$ to get the threshold value, h . The following change-point decision logic is sufficient to detect a change:

```

IF  $g(r(i)) \geq h$  THEN
    Parameter change has occurred
ELSE
    no change-point
END IF

```

3.4.1.6 Offline Implementation of the Change-Point Detection with NLSSA

The time delay and embedding dimension does not change. The trained neural network model behaves like a parametric model. As long as there is no change in the parameters of the system that generate the time series, the model will generate residuals that have more or less equal magnitude. The embedding lag and embedding dimension remains constant for the system. The threshold value calculated when the model was trained also remains constant.

- Embed time series using the same embedding lag and embedding dimension that was used during model training.

Let $x(i) = x_1, x_2, \dots, x_N$ be a time series, with N data points

Let embedding lag be τ and embedding dimension be d

$$X(i) = \begin{pmatrix} x_n & x_{n+1} & x_{n+2} & \cdots & x_{N-(d-1)\tau} \\ x_{n+\tau} & x_{n+1+\tau} & x_{n+2+\tau} & \cdots & x_{N-(d-2)\tau} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n+(d-1)\tau} & x_{n+(d-1)\tau+1} & x_{n+(d-1)\tau+2} & \cdots & x_N \end{pmatrix} \quad (3.41)$$

- Use model to simulate results for the new time series,

$$Y(i) = \text{sim}(M_{FF}(\theta), X(i)) \quad (3.42)$$

$$Y(i) = \begin{pmatrix} y_n & y_{n+1} & y_{n+2} & \cdots & y_{N-(d-1)\tau} \\ y_{n+\tau} & y_{n+1+\tau} & y_{n+2+\tau} & \cdots & y_{N-(d-2)\tau} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{n+(d-1)\tau} & y_{n+(d-1)\tau+1} & y_{n+(d-1)\tau+2} & \cdots & y_N \end{pmatrix} \quad (3.43)$$

- Unembed $Y(i)$ to get the simulated time series

$$y(i) = y_1, y_2, \dots, y_N \quad (3.44)$$

- Calculate residuals and square them,

$$r(i)^2 = (x(i) - y(i))^2 \quad (3.45)$$

- Calculate 95% or 99% quantile $r(i)^2$ to get the threshold value, h
- Plot residuals in the same graph with the threshold (h).
- To minimise false alarms, generate a new residual series by doing moving average on residual series. This will make the residual time series smooth and thus minimise false change-point detection. In this case, the moving average window length becomes important as it will have an effect on the change-point detection delay time.
- Compare the residuals with the threshold (h).

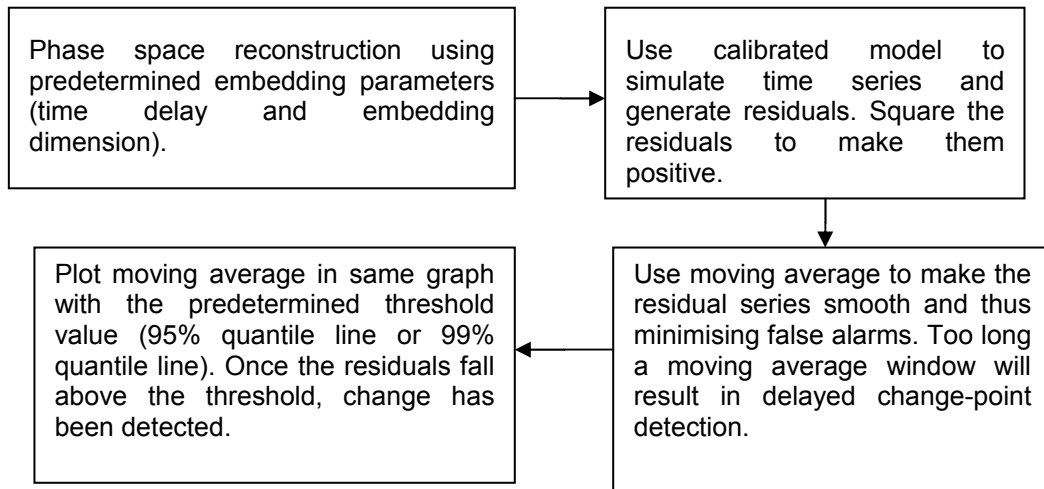


Figure 3.11: Offline change-point detection scheme.

3.4.1.7 Online Implementation of the Change-Point Detection with NLSSA

The model is trained or calibrated offline. The embedding parameters remain constant for the same system and model.

- Sample time series at a time interval. The time interval should be long enough to have generated a suitable time series to enable embedding and change-point detection. The time interval will also depend on the SCADA sampling interval.
- Put the time series through the change-point detection algorithm.

The algorithm performs the following calculations:

- Embeds the time series using embedding parameters calculated during model development (equation 3.41);
- Uses the calibrated model to simulate output, generate residuals and square them (equation 3.42 – 3.45);
- Uses the CUSUM algorithm to evaluate the residuals.

The CUSUM algorithm is used because the residuals can be regarded as stochastic. With the knowledge of stochastic processes, the mean value of the residuals in the 'in control' region must be different from the mean value of the residuals when there has been a parameter change. The CUSUM algorithm just utilizes the characteristic of stochastic signals such as residuals to implement the detection of change-points.

The CUSUM supplies a decision function, marked as g_i and a threshold h specified beforehand. If $g_i \leq h$, then there is no change-point, otherwise when $g_i \geq h$, there has been a change in parameters. The value of the threshold is calculated during model calibration. The value of h determines the rate of false alarms and the chance of missing the change-point. The smaller the value of h the higher the rate of false alarms and a big value of h means fewer false alarms but a higher chance of missing the change-point. The details of the CUSUM test and algorithm can be found in Appendix A.

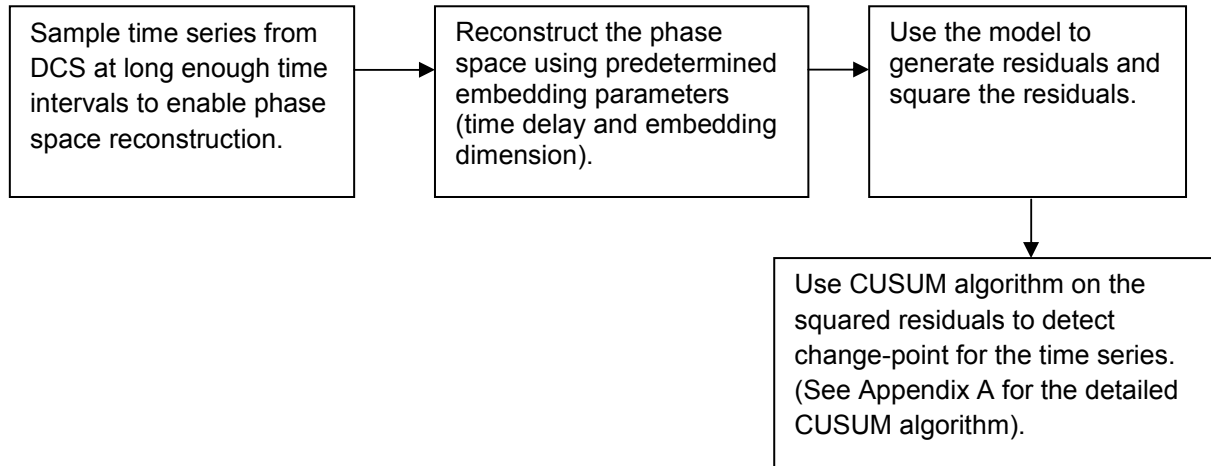


Figure 3.12: Online change-point detection scheme.

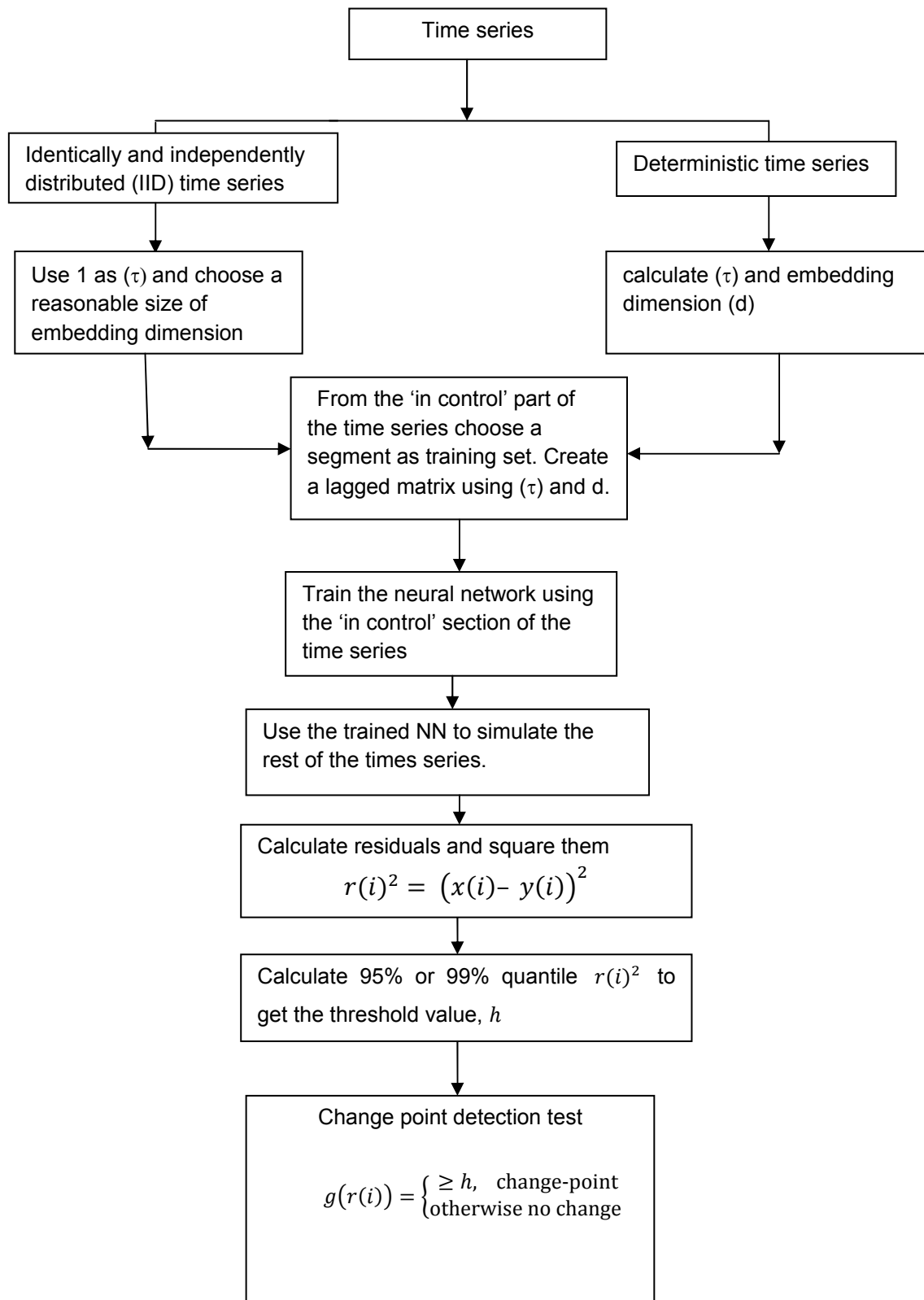


Figure 3.13: Summary of change point detection algorithm.

Chapter 4 Case Studies

Three case studies are considered in this work. The first consists of four Gaussian data time series with different properties, namely T_1 , T_2 , T_3 , and T_4 . All these are linear time series. The second case study is the BZ reaction system and the third is the autocatalytic reaction system with fourth being the Lotka-Volterra Model.

4.1 Gaussian Time Series Data

In the first case study, four systems (T_1 , T_2 , T_3 , and T_4) are considered. The first three ($T_1 - T_3$) represent univariate time series observations and the fourth (T_4) multivariate time series observations. The systems are briefly described below.

- T_1 : 500 samples, of which the first 250 are identically, independently distributed Gaussian data, with zero mean and unit variance. The last 250 samples have a mean of 2 and unit variance to simulate an abrupt mean shift in the data.
- T_2 : 500 samples, of which the first 250 are identically, independently distributed Gaussian data, with zero mean and unit variance. The last 250 samples have a mean of 0 and a variance of 2 to simulate an abrupt shift in the variance of the data.
- T_3 : 500 samples of data generated by an autoregressive process of the form $x(t+1) = fx(t) + e(0,0.1)$, where e has a zero mean Gaussian distribution with a variance of 0.1. In the first 250 samples, parameter $f = 0.9$ and in the 2nd 250 samples, $f = 0.5$ to simulate an abrupt shift in the autocorrelation of the data.
- T_4 : Two cross-correlated time series of 500 samples each. The first 250 time series samples have a covariance matrix of

$$\mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

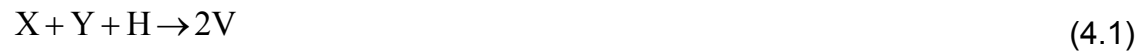
while the last 250 observations have a covariance matrix of

$$\mathbf{C} = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

This simulates an abrupt shift in the correlation between the variables.

4.2 Belousov-Zhabotinsky (BZ) reactions

The Belousov-Zhabotinsky (BZ) reaction is an unstable chemical reaction that maintains self-oscillations and propagating waves, which may display chaos under certain conditions. A simplified model by Györgyi and Field (1991) is considered here. It has the following chemical scheme (Zhang et al., 1993):



In (4.15)-(4.11), $A = \text{BrO}_3^-$, $H = \text{H}^+$ and $M = \text{MA}$ are chemical species with constant concentrations, while the concentrations of $Y = \text{Br}^-$, $X = \text{HBrO}_2$, $Z = \text{Ce(IV)}$ and $V = \text{BrMA}$ are the model variables. The rates (r) and rate constants (c) are (with $[\cdot]$ indicating the concentration of the relevant reactant):

$$r_1 = c_1[H][X][Y], \quad c_1 = 4.0 \times 10^6 \quad (4.8)$$

$$r_2 = c_2[A][H]^2[Y], \quad c_2 = 2 \quad (4.9)$$

$$r_3 = c_3[X]^2, \quad (4.10)$$

$$r_4 = c_4[A]^{0.5}[H]^{1.5}(C-[Z])[X]^{0.5}, \quad c_4 = 55.2 \quad (4.11)$$

$$r_5 = c_5[X][Z], \quad c_5 = 7000 \quad (4.12)$$

$$r_6 = gc_6[Z][V], \quad c_6 = 0.09 \quad (4.13)$$

$$r_7 = \beta c_7[M][Z], \quad c_7 = 0.23 \quad (4.14)$$

By assuming that Y is a fast variable, the system of scaled differential equations is:

$$\begin{aligned} \frac{dx}{d\mu} = T & \left[-c_1 H Y_0 x \tilde{y} + \frac{c_2 A H^2 Y_0}{X_0} \tilde{y} - 2c_3 X_0 x^2 + \dots \right. \\ & \left. \frac{1}{2} c_4 A^{0.5} H^{1.5} X_0^{-0.5} (C - Z_0 z) x^{0.5} - \frac{1}{2} c_5 Z_0 x z - c_f x \right] \end{aligned} \quad (4.15)$$

$$\frac{dz}{d\mu} = T \left[c_4 A^{0.5} H^{1.5} X_0^{0.5} \left(\frac{C}{Z_0} - z \right) x^{0.5} - c_5 X_0 x z - gc_6 V_0 z v - hc_7 M z - c_f x \right] \quad (4.16)$$

$$\frac{dv}{d\mu} = T \left[\frac{2c_1 H X_0 Y_0}{V_0} x \tilde{y} + \frac{c_2 A H^2 Y_0}{V_0} \tilde{y} + \frac{c_3 X_0^2}{V_0} x^2 - gc_6 Z_0 z v - c_f v \right] \quad (4.17)$$

where $\mu = t / T$ and $T = (10c_2 A H C)^{-1}$ are the scaled time with scaling factors: $x = X/X_0$, $X_0 = c_2 A H^2 / c_5$, $Y_0 = 4c_2 A H^2 / c_5$, $z = Z/Z_0$, $Z_0 = CA/(40M)$, and $v = V/V_0$, and $V_0 = 4AHC/M^2$, is the scaled concentration variables.

4.3 Autocatalytic Process

The autocatalytic process consists of two parallel, isothermal autocatalytic reactions taking place in a continuous stirred tank reactor (CSTR) (Lee & Chang, 1996). The system is capable of producing self-sustained oscillations based on cubic autocatalysis with catalyst decay at certain parameters. The system proceeds mechanically as follows:





where A, B and C are participating chemical species. The reaction rates are governed by the following rate equations:

$$-r_A = k_1 c_A c_B^2 \quad (4.21)$$

$$r_C = k_2 c_B \quad (4.22)$$

$$-r_D = k_3 c_D c_B^2 \quad (4.23)$$

where k_1 , k_2 and k_3 are the rate constants for the chemical reactions and c_A , c_B and c_D are concentrations of species A, B and D respectively. To describe the system using three ordinary differential equations, the dimensionless concentration, ratios of species in feed and dimensionless time are described as follows:

$$X = \frac{C_A}{C_{A,0}}, \quad Y = \frac{C_D}{C_{D,0}}, \quad Z = \frac{C_B}{C_{B,0}} \quad (4.24)$$

$$a = \frac{k_1 C_{B,0}^2 V}{Q}, \quad b = \frac{k_3 C_{B,0} V}{Q}, \quad c = \frac{k_2 V}{Q} \quad (4.25)$$

$$\gamma_1 = \frac{C_{A,0}}{C_{B,0}}, \quad \gamma_2 = \frac{C_{D,0}}{C_{B,0}} \quad (4.26)$$

$$\tau = \frac{t.Q}{V} \quad (4.27)$$

The ordinary differential equations governing the system are subsequently defined by

$$\frac{dX}{dt} = 1 - X - aXZ^2 \quad (4.28)$$

$$\frac{dY}{dt} = 1 - Y - bYZ^2 \quad (4.29)$$

$$\frac{dZ}{dt} = 1 - (1 + c)Z + \gamma_1 aXZ^2 + \gamma_2 bYZ^2 \quad (4.30)$$

Lee and Chang (1996) have shown that for $a = 18000$; $b = 400$; $c = 80$; $\gamma_1 = 1.5$; $\gamma_2 = 4.2$ and with initial conditions $X_0 = 0$; $Y_0 = 0$; $Z_0 = 0$, the system exhibits chaotic behaviour.

The system (4.28)-(4.30) was simulated with the ODE45 subroutine in MATLAB 7.2 at a sampling rate of 0.005. For the first 10 000 observations the same parameters as specified by Lee and Chang (1996) were used. A simulated fault condition was then introduced through a slight increase in the feed concentrations of A and D . This caused an increase in the parameters γ_1 and γ_2 . As with the previous two cases, the parameters γ_1 and γ_2 were allowed to drift slowly from their starting values, $\gamma_1 = 1.5$ and $\gamma_2 = 4.2$, to $\gamma_1 = 1.55$ and $\gamma_2 = 4.25$, after which the parameters were again kept constant at their new values, $\gamma_1 = 1.55$ and $\gamma_2 = 4.25$, for another 10 000 observations. The 30 000 observation were then sampled to get 3 000 index points that have equal sampling intervals. The first 1000 points have parameters $\gamma_1 = 1.5$ and $\gamma_2 = 4.2$ and the points 1001 to 2000 have the parameters γ_1 and γ_2 slowly ramping up from 1.5 and 4.2 to 1.55 and 4.25 respectively. The last segment of 2001 to 3000 points have $\gamma_1 = 1.55$ and $\gamma_2 = 4.25$.

Table 4-1 parameter changes for the autocatalytic process

Data range	γ_1	γ_2
0 – 10 000	1.5	4.2
10 001 – 20 000	1.5 – 1.55	4.2 – 4.25
20 001 – 30 000	1.55	4.25

4.4 Lotka-Volterra Model

There is an increase in the use of biological processes in the field of chemical engineering. These include amongst others, fermentation processes, extraction of low grade mineral ores using bacteria, treatment of waste products using aerobic processes, etc. Biological processes are widely used in agriculture where certain organisms are introduced to control other organisms that are harmful to crops. Sometimes insecticides or pesticides are used to control certain harmful organisms, this leads to a disturbance in the ecological balance of the predator-prey system.

The Lotka-Volterra model is a predator-prey model based on the natural way of species survival.

Biologically, it is natural to study the existence and asymptotical stability of equilibria and limit cycles for autonomous predator-prey systems with these functional responses or general interactions of Kolmogorov type (Liu & Chen, 2003)). Without exception, biological communities are visited by perturbations that occur in a more or less periodic fashion. Thus an interesting problem arises from forcing predator-prey systems periodically and observing the change in the oscillation behaviour.

The Lotka-Volterra model is the simplest model of predator-prey interactions, developed independently by Lotka (1925) and Volterra (1926).

$$\frac{dH}{dt} = rH - aHP \quad (4.31)$$

$$\frac{dP}{dt} = bHP - mP \quad (4.32)$$

It has two variables, P and H , and several parameters, r, a, b and m , where H = density of prey, P = density of predators, r = intrinsic rate of prey population increase, a = predation rate coefficient, b = reproduction rate of predators per 1 prey eaten, and m = predator mortality rate. The model is a nonlinear system that is not able to exhibit chaotic behaviour since it has only two variables. The system oscillates when the parameters are set at $r = 2$, $a = 0.001$, $b = 10$ and $m = 0.002$.

Consider a situation where the environment changes to favour 'b', the reproduction rate of predators per 1 prey eaten. For the biological system to be sustainable, this should trigger changes in other parameters to counter the increase in 'b', i.e. 'r' can increase and result in the rapid production of prey, or the mortality of predators can increase (increase in m). Thus, in this system, the parameters change as follows:

Table 4-2: Parameter changes for Lotka–Volterra model

Parameter	Initial	Final
r	2	3
b	10	11
a	0.001	0.0011
m	0.002	0.0021

The system was simulated with ODE 45 subroutine in MATLAB 7.2 with a sampling rate of 0.02 seconds. The initial values of P and H were set at 100 and 500 respectively. The parameters were kept constant at their initial values for the first 10 000 observations. The dynamic change was introduced at the 10,000th observation. The parameters were increased stepwise over the next 10 000 observations. The increase was effected after every 1 000 observations. The parameters were again kept constant from samples 20,001 to the 30,000.

Chapter 5 Results and Discussion

An auto-associative neural network was used to detect change-points for the time series described in Chapter 4. A model of the process that generates the data series is developed using the first section of the time series before a parameter change is introduced. When the model is used to reproduce the time series, it generates residuals as it is not able to reproduce the exact values of the time series (see Figure 3.10). When the parameter change is introduced, the residuals generated by the model are expected to increase as the parameters of the modelled system change. $e_i = x_i - \hat{x}_i$. The residuals are first squared before a test statistic is used to determine the change-point according to the square of residuals, i.e., $e_i e_i^T = (x_i - \hat{x}_i)^2$.

A 95% quantile is used as a threshold value. It is calculated on the residuals before the parameter change; any points above the threshold have a high chance of being a result of a parameter shift.

The change-point detection using neural networks is compared with change-point detection method based on singular spectrum analysis (SSA). The fraction of explained variance (FEV) in nonlinear singular spectrum analysis (NLSSA) is similar in spirit to the eigenvalues of the covariance matrix that explains the energy percentage captured by linear singular spectrum analysis (LSSA).

$FEV = \frac{E[\|\hat{x}(t)\|^2]}{E[\|x(t)\|^2]}$, FEV from NLSSA will be compared with eigenvalues for LSSA.

A program developed by Moskvina & Zhigljavsky, 2003 was used for change-point detection. The algorithm used in the program is discussed in detail in their publication (Moskvina & Zhigljavsky, 2003).

5.1 Gaussian Time Series Data

Figure 5.1 is a plot of Gaussian data (T1) with a variance of 1. The first 250 points of the Gaussian series has a mean of 0 and the last 250 points have a mean of 2. Figure 5.1 shows two series - the original series and the simulated series. A neural network with three mapping and three de-mapping layers and two neurons at the

bottleneck was trained. The trained network was then used to generate the values of the time series, denoted SimulateT1.

Figure 5.2 is a plot of Gaussian data (T2) with a mean of 0. The first 250 points have a variance of 1 and the last 250 points have a variance of 2. The first 250 points of both series (T1 and T2) are the same as they both have mean of 0 and variance of 1. Therefore the same network trained for T1 was used to simulate the simulated series, denoted Simulate T2.

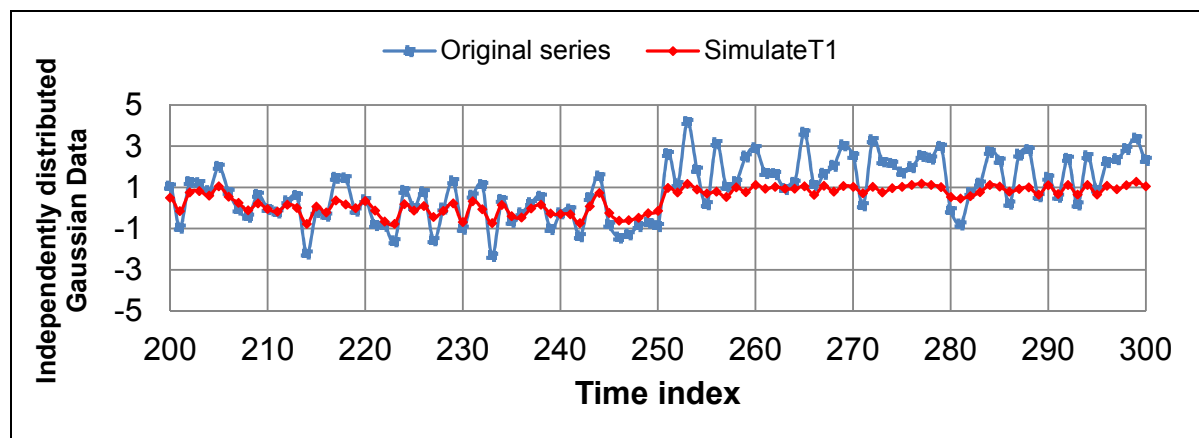


Figure 5.1: Plot of independently distributed Gaussian data with a mean step change from 0 to 2.

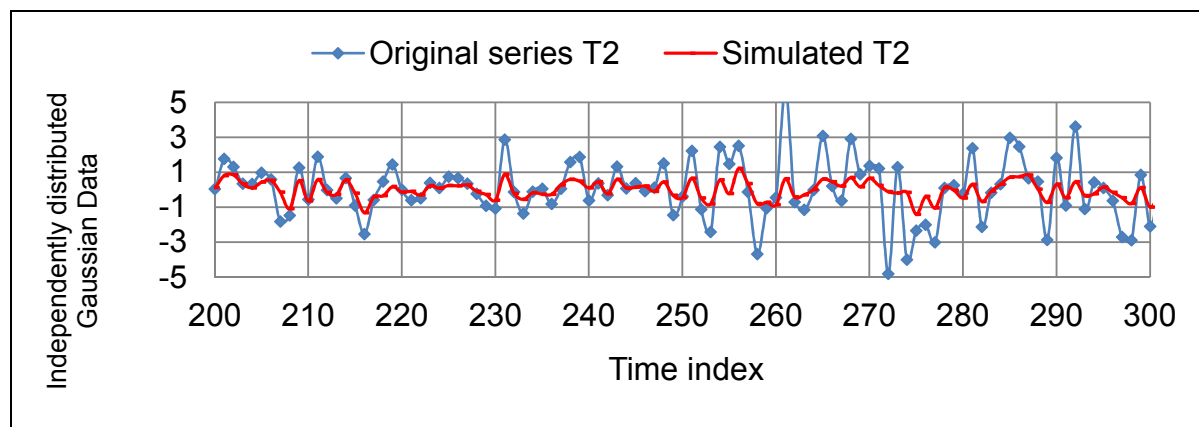


Figure 5.2: Plot of independently distributed Gaussian data with a variance step change from 1 to 2.

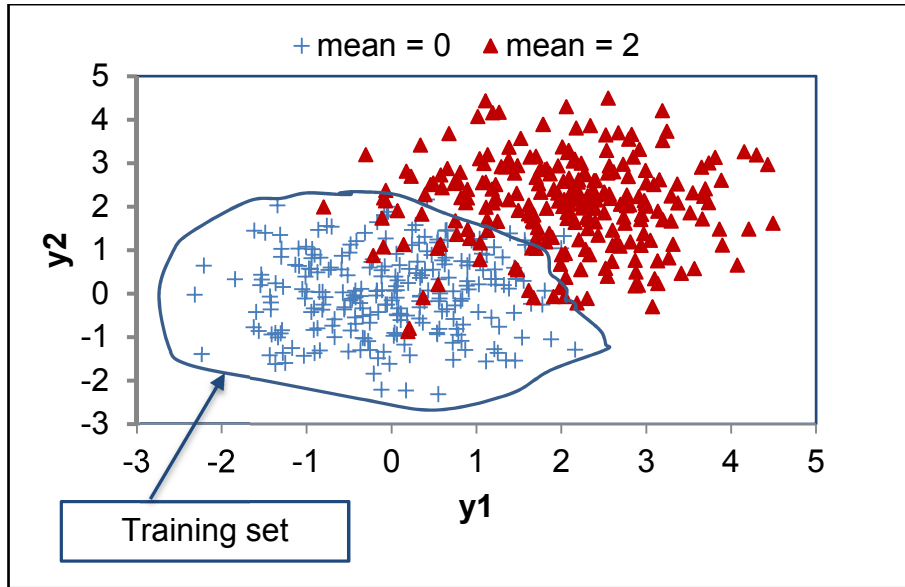


Figure 5.3: Scatter plot of Gaussian data with a variance of 1 and a step change of mean values from 0 to 2.

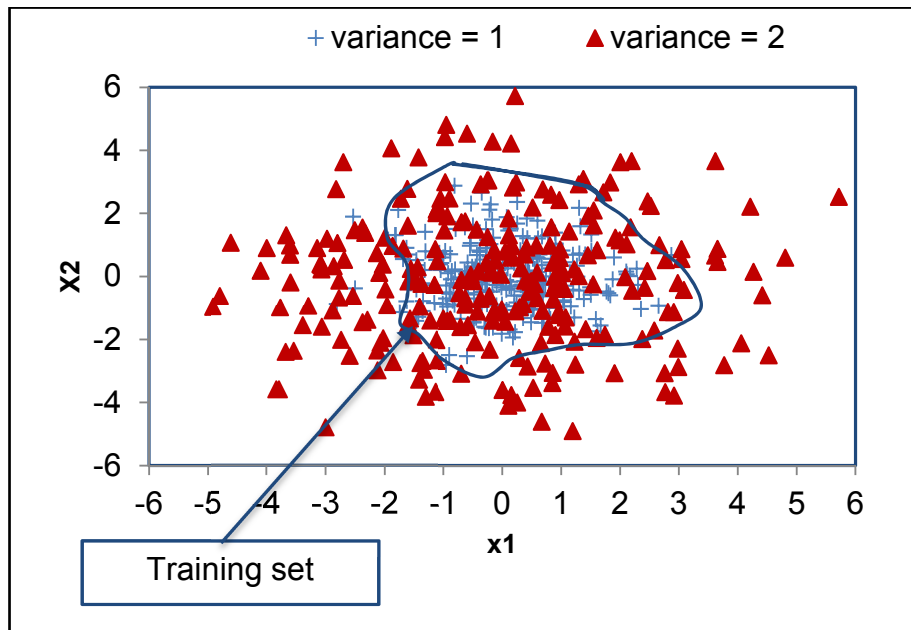


Figure 5.4: Scatter plot of Gaussian data with a mean of 0 and a step change of variance from 1 to 2

Figures 5.3 and 5.4 show scatter plots of the time series T1 and T2. The encircled points are the first 250 points and the training data set is taken from them. The demarcation is clearly defined in T1 with a mean shift from 0 to 2 (Figure 5.3), whereas with the shift in variance, the training set is embedded inside the second set that has a variance of 2 (Figure 5.4).

5.1.1 Network Training

The first 150 points were used for training the network – this training set is encircled in Figures 5.3 and 5.4. Lagged copies of the training set were used as the input to the network. An embedding dimension of 5 and lag time of 1 were used. Ten runs of each network were done and the one that showed the best performance was chosen. This was necessitated by the tendency of the neural network to be trapped in local minima.

Table 5.1 below shows the training results. It shows that the number of neurons in the mapping and de-mapping layers does not affect the performance of the network much, see also Figure 5.6. The network performance (MSE) and fraction of explained variance (FEV) does not improve much with increase in the number of neurons for the mapping and de-mapping layers. The number of neurons is related to the complexity of the nonlinear functions that can be generated by the network. Increasing the number of neurons for the bottleneck both improves the network performance (MSE decreases) and increases the fraction of explained variance, see Figures 5.5 a) and 5.5 b) below.

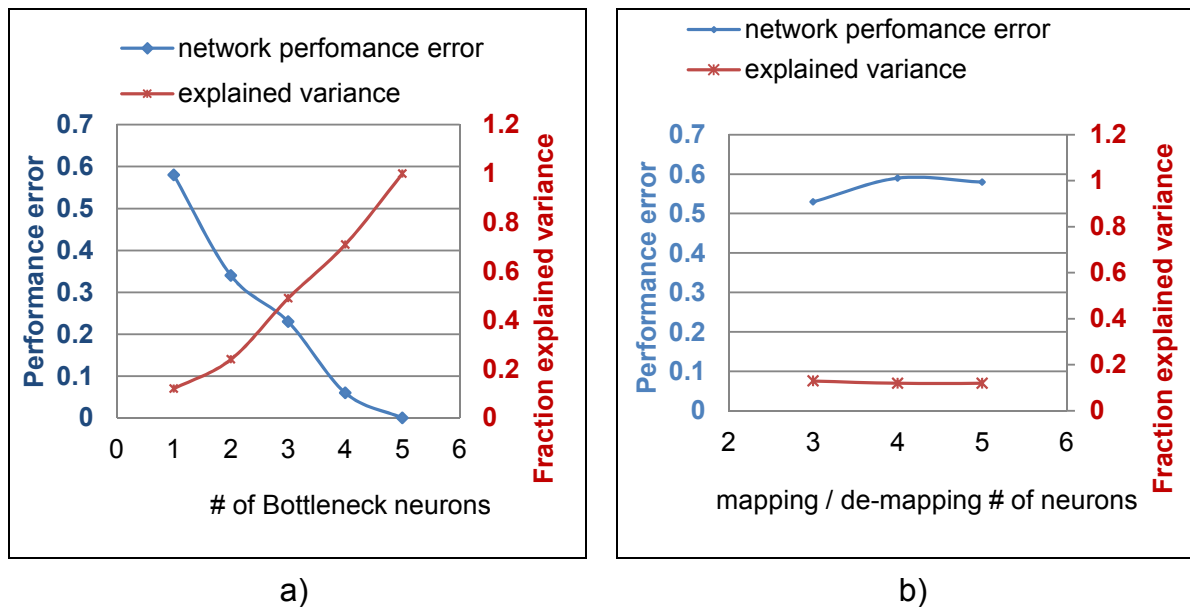


Figure 5.5: a) Variation of network performance (MSE) and FEV with an increase in number of bottleneck neurons, b) Variation of network performance MSE and FEV with increase number mapping/de-mapping neurons

Table 5-1 : Shows training results for different network sizes.

Network Structure			MSE error	Regression			FEV
Mapping layer	Bottleneck layer	De-mapping layer		Training	Validation	Testing	
3	1	3	0.53	0.58	0.48	0.46	0.13
4	1	4	0.59	0.55	0.47	0.42	0.12
5	1	5	0.58	0.61	0.52	0.55	0.12
3	2	3	0.43	0.71	0.59	0.57	0.24
4	2	4	0.33	0.78	0.63	0.58	0.23
5	2	5	0.34	0.79	0.61	0.58	0.24
5	3	5	0.23	0.86	0.75	0.83	0.49
5	4	5	0.06	0.97	0.93	0.90	0.71
5	5	5	0.00	1.00	1.00	1.00	1

Note: Fraction of explained variance was only done on the training set, i.e. before the step change.

Data points not used in the network training but still in the batch before the parameter change were used when determining the explained variance.

Gaussian data is not correlated and as expected, if it is embedded with a dimension of 5, it will need five neurons in the bottleneck to explain 100% of the variance. Figure 5.5 a) shows a plot of network performance (MSE) and fraction of explained variance (FEV) with the increase of the bottleneck. FEV reaches 100% when the number of neurons in the bottleneck equals the embedding dimension. Figure 5.5 b) shows that there is a minimal effect when increasing the mapping and de-mapping nodes on the performance of the network.

5.1.2 Change-Point Detection of Gaussian Data (T1 and T2)

An auto-associative neural network model was developed as described in section 3.4.1 (network training). The network was used to simulate the rest of the data in T1 and T2. The squared difference between the expected values and the network generated values was calculated and plotted in Figures 5.6 and 5.7 below. A 95%

quantile of the residuals generated from the data points before the parameter change was calculated and plotted on the same graph as the residuals. The change-point is said to have occurred once the residuals are more than the threshold value, which is the 95% quantile.

5.1.2.1 Nonlinear Singular Spectrum Analysis

Figures 5.6 and 5.7 below show that the change-point happened at time index 250, which is in agreement with the data.

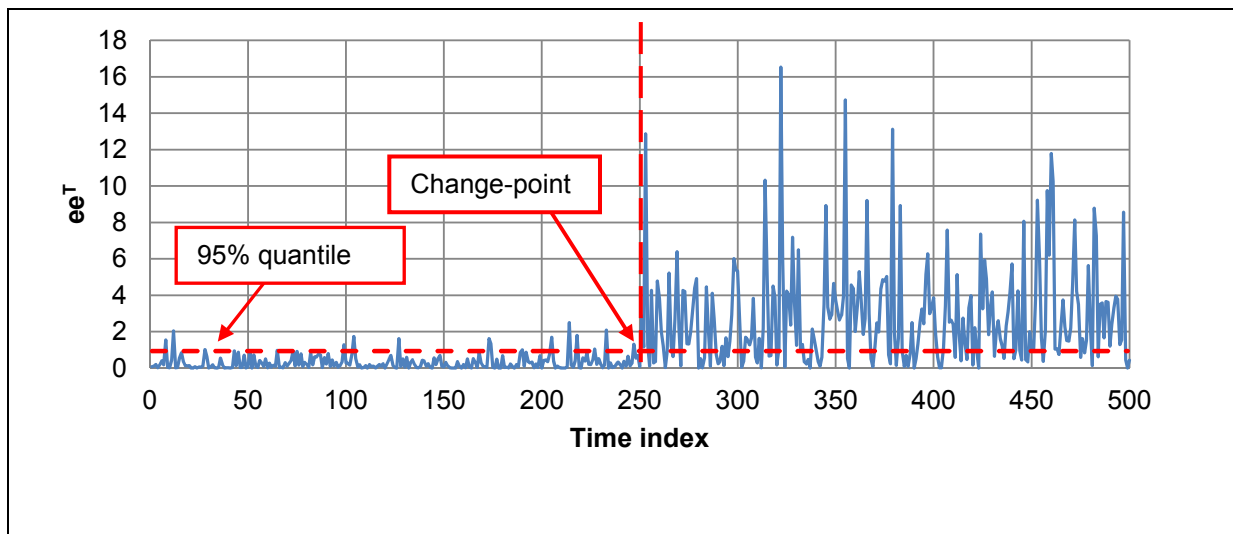


Figure 5.6: Change-point detection of independently distributed Gaussian data with a mean step change from 0 to 2 using a neural network.

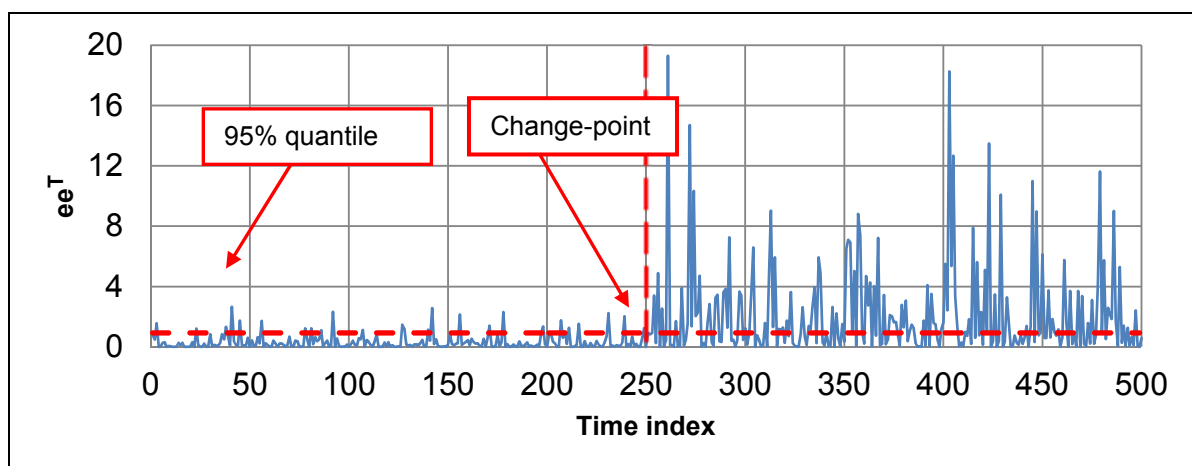


Figure 5.7: Change-point detection of independently distributed Gaussian data with a variance step change from 1 to 2 using a neural network.

5.1.2.2 Linear Singular Spectrum Analysis (LSSA)

Singular spectrum analysis was also used to detect change on T1 and T2. A program developed by Moskvina and Zhigljavsky (2003), was used and is discussed in detail in Chapter 3.2. Figures 5.8 and 5.9 show the results of the change-point detection using SA on the sets T1 and T2 respectively. The change-point is detected at time index 250 for T1 (Figure 5.8), and at time index 260 for T2. The detection had a delay of ten time index units.

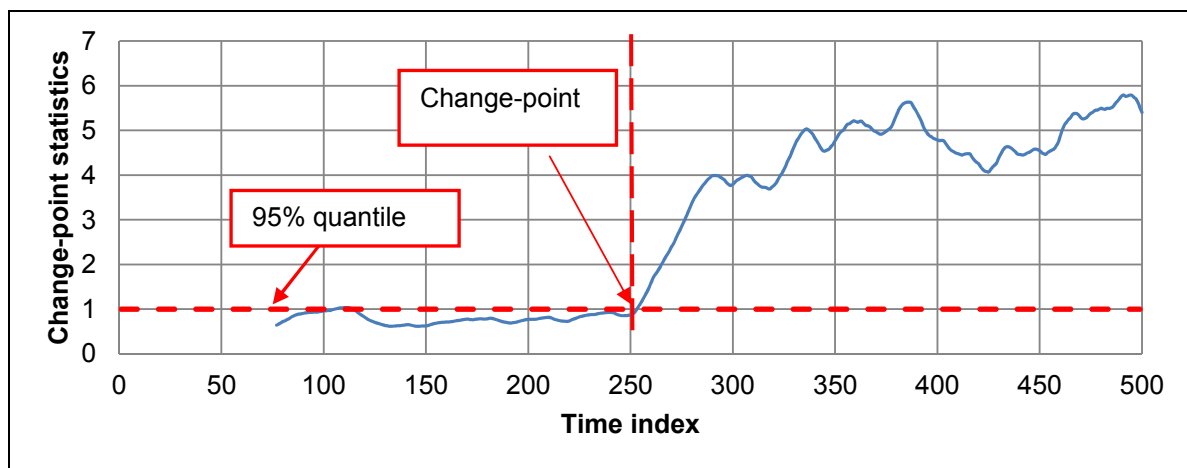


Figure 5.8: Change-point detection with SSA for Gaussian data that has a mean step change from 0 to 2.

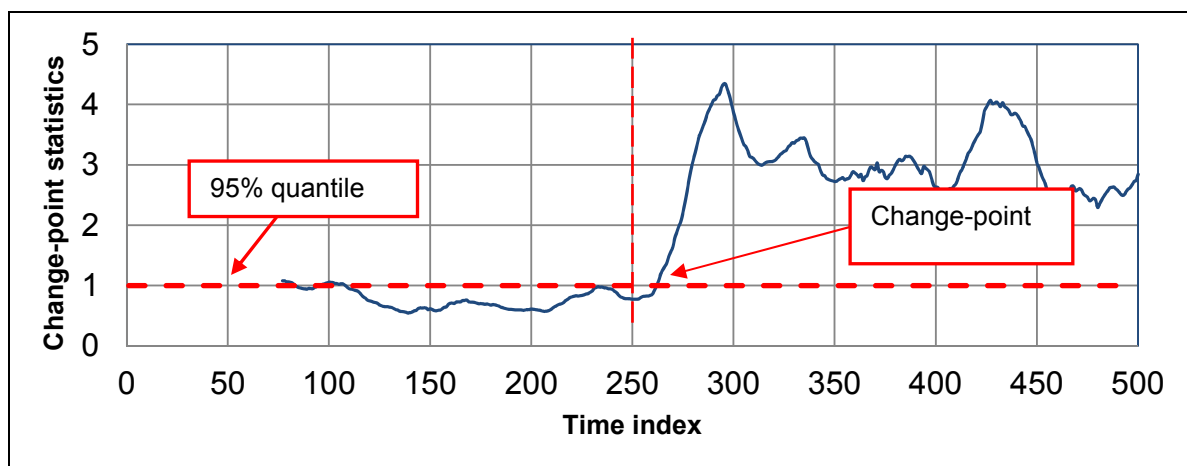


Figure 5.9: Change-point detection with SSA for Gaussian data that has a variance step change from 1 to 2.

In series T1 the mean was increased from 0 to 2 whilst the variance stayed constant at 1. In series T2 the variance was increased at time index 250 from 1 to 2 whilst the mean remained constant at 0. The change-point is defined as a parameter shift – either an increase or decrease. For both T1 and T2, the shift is a parameter increase. Figures 5.10 and 5.11 show the change-point when the parameter shift is a decrease. For T1 the training set was taken from the batch with a mean of 2 and the parameter shift is from mean = 2 to mean = 0 and the variance remained constant. For T2 the training set was sampled from the batch with variance 2 and the parameter shift was from 2 to 1.

The results of the change-point detection are shown in Figures 5.10 and 5.11. The mean shift from 2 to 0 is well detected, as shown by Figure 5.10. An observation of Figure 5.3 shows that the scatter plot for mean step change is distinctly segregated; therefore training using any of the batches should be able to show the change-point.

Figure 5.11 shows that a decrease in the variance from 2 to 1 cannot be picked up by the change-point statistics since it falls within the limit. Figure 5.4 also shows that the batch that has variance = 1 is a subset of the batch with variance = 2. This also follows with the residuals and thus change-point cannot be detected.

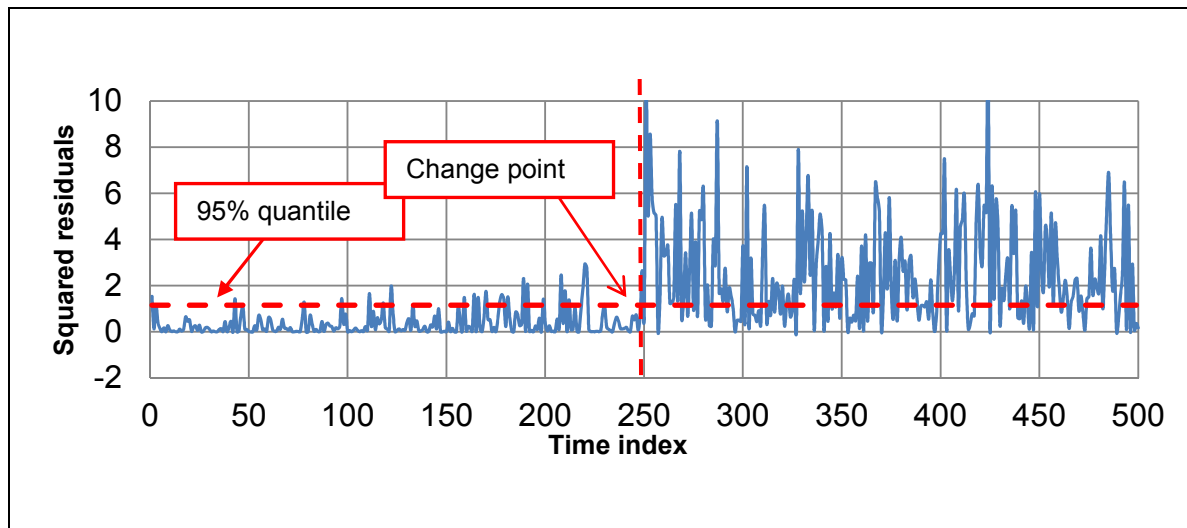


Figure 5.10: Change-point detection of independently distributed Gaussian data with a mean step change from 2 to 0 using NLSSA.

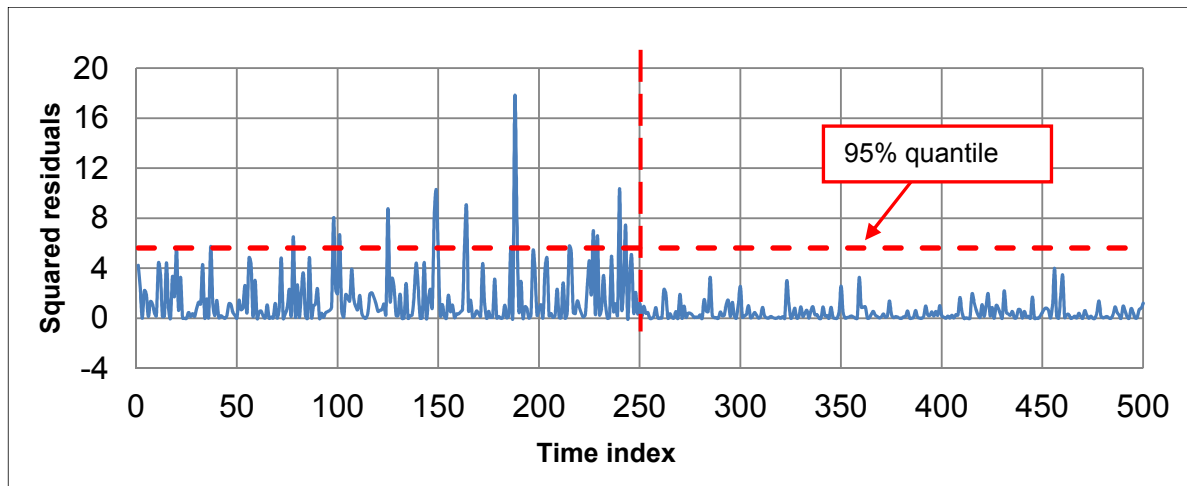


Figure 5.11: Change-point detection of independently distributed Gaussian data with a variance step change from 2 to 1 NLSSA

5.1.2.3 Comparison between NLSSA and LSSA on FEV

Fraction of explained variance (FEV) in neural networks can be equated to eigenvalues in linear singular spectrum analysis or principal component analysis. Table 5.2 below shows eigenvalues from the decomposition of the T1 and T2 series as a percentage, as explained above.

Table 5-2: A comparison of FEV from neural networks and FEV from LSSA.

Eigennumber	FEV NLSSA ,T1 & T2	FEV LSSA ,T1 and T2
1	0.12	0.75
2	0.24	0.82
3	0.49	0.88
4	0.71	0.94
5	1	1

LSSA performs better than NLSSA on the Gaussian data with mean and variance shifts. Table 5.2 shows that the first principal component for NLSSA explains 12% of the variance in both T1 and T2, whereas LSSA explains 75.1% for T1 and T2. Both NLSSA and LSSA detect the change-point perfectly for the Gaussian data, in Figures 5.6 to 5.9.

5.2 Autoregressive Process

Figures 5.12 to 5.13 show the change-point detection of a step change samples of data generated by an autoregressive process of the form $\mathbf{x}(t+1) = f(\mathbf{x}(t)) + \mathbf{e}(0,0.1)$, where \mathbf{e} has a zero mean Gaussian distribution with a variance of 0.1. In the first 250

samples, parameter $f = 0.9$ and in the second 250 samples, $f = 0.5$ to simulate an abrupt shift in the autocorrelation of the data.

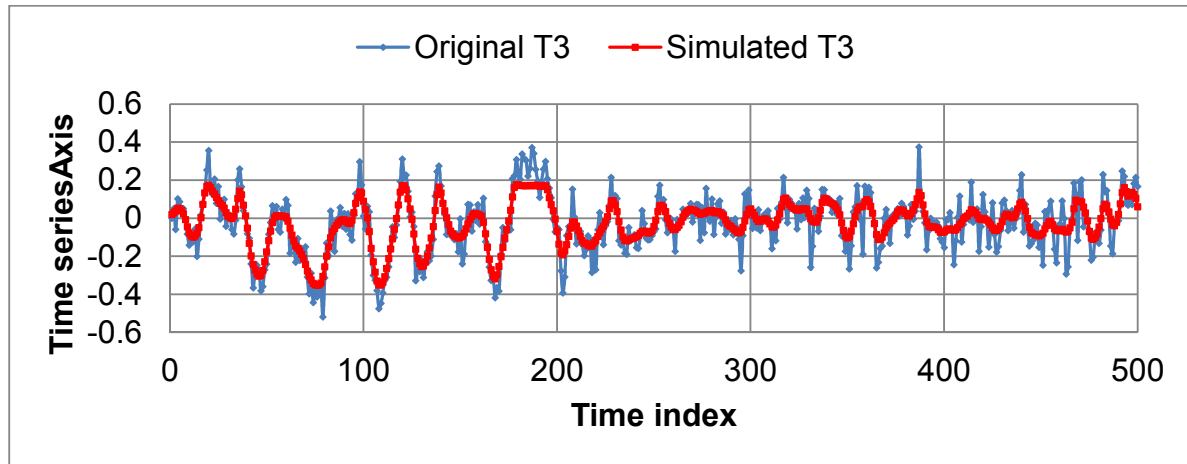


Figure 5.12: Time series for an autoregressive process of the form $x(t+1) = f(x(t)) + e(0,0.1)$, where e has a zero mean Gaussian distribution with a variance of 0.1, and a step change in f from 0.9 to 0.5

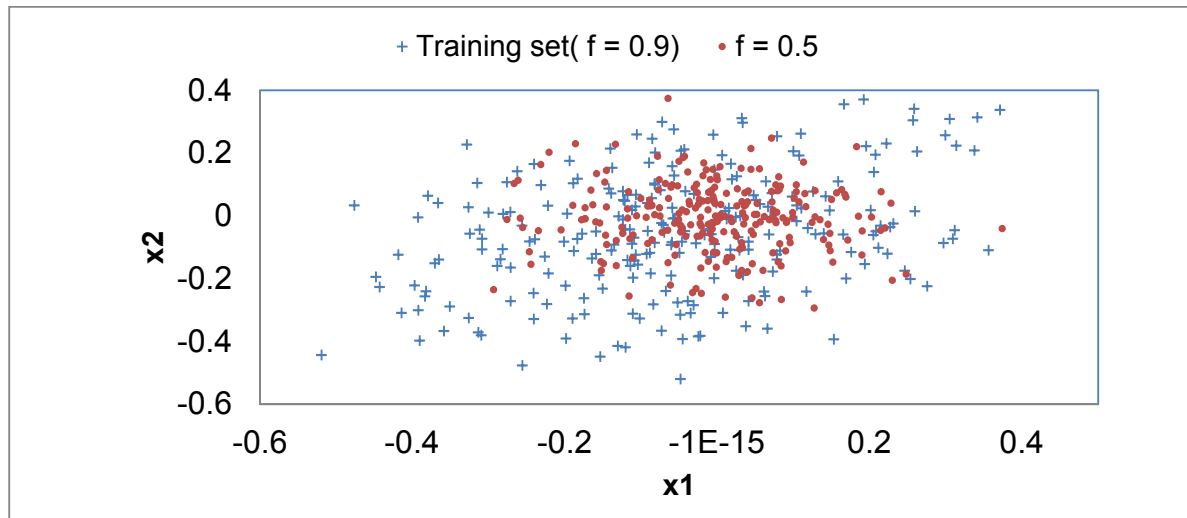


Figure 5.13: Scatter plot of time series generated by an autoregressive process of the form $x(t+1) = f(x(t)) + e(0,0.1)$, with a step change of f from 0.9 to 0.5.

5.2.1 Network Training

The first 150 points were used for training the network. Lagged copies of the training set were used as the input to the network. An embedding dimension of 5 and lag time of 1 were used. Ten runs of each network were done and the one that showed the best performance was chosen.

Table 5.3 below shows the training results. The number of neurons in the mapping and de-mapping layers does not have much of an impact on the performance of the

network (see also Figure 5.14 a)). The network performance (mean square error) and fraction of explained variance does not improve much with an increase in the number of neurons for the mapping and de-mapping layers. Increasing the number of neurons for the bottleneck improves the network performance (mean square error decreases) and also increases the fraction of explained variance, see Figure 5.14 b) below.

Table 5-3: Training results for different network sizes.

Network structure			Network performance	Regression			FEV
Mapping layer	Bottleneck layer	De-mapping layer	MSE error	Training	Validation	Testing	
3	1	3	0.0078	0.90	0.88	0.85	0.65
4	1	4	0.0062	0.90	0.86	0.83	0.65
5	1	5	0.0075	0.89	0.82	0.89	0.65
3	2	3	0.0024	0.97	0.95	0.95	0.91
4	2	4	0.0025	0.96	0.95	0.96	0.90
5	2	5	0.0028	0.95	0.97	0.97	0.89
5	3	5	0.0014	0.98	0.97	0.98	0.93
5	4	5	0.0006	0.99	0.99	0.83	0.99

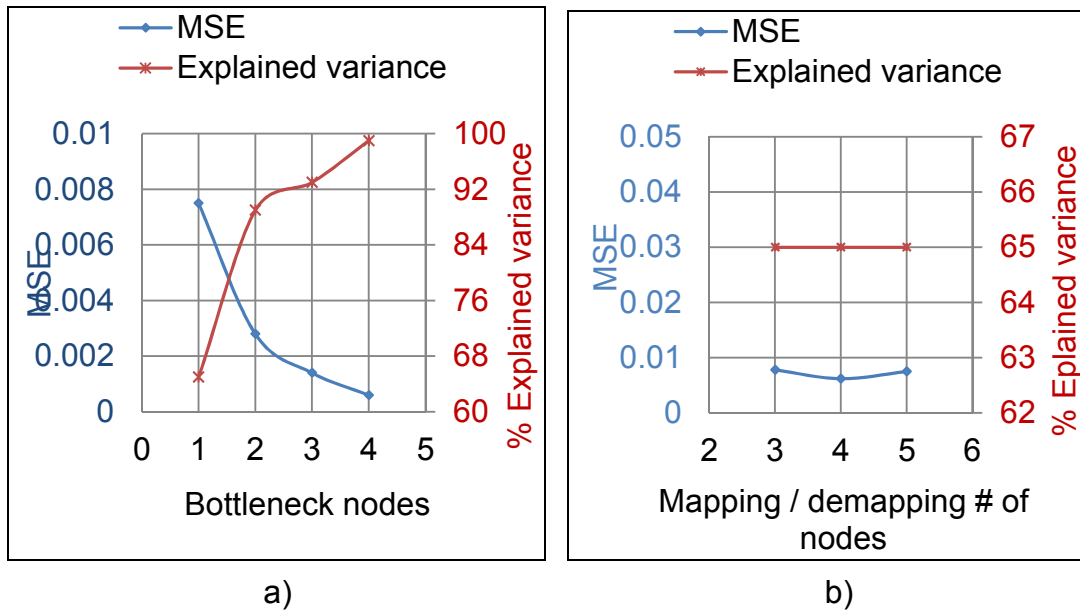


Figure 5.14: a) shows the behavior of the network when increasing the number of neurons in the bottleneck, b) shows the behavior of the network when increasing the number of neurons of the mapping/de-mapping layer.

5.2.2 Change-Point Detection Using Nonlinear Singular Spectrum Analysis

Change-point detection was done on the autoregressive process as described in section 4.4.1. Figure 5.15 shows the change-point detection of the autoregressive process. NLSSA fails to detect the change that was introduced at time index 250. NLSSA shows peaks even before the change was introduced. The behaviour of the residual space is the same before and after the parameter change. The developed model is able to model the time series accurately even after the parameter change. This results in the residuals being of the same magnitude throughout and hence the failure to detect the change-point. A time series with the same parameters was developed: the developed time series started with a parameter of $f = 0.5$ and at index 250 f was increased to 0.9. The aim was to check whether change detection would be possible when the parameter change was an increase instead of a decrease. The change-point detection is plotted in Figure 5.19, again the change-point could not be detected.

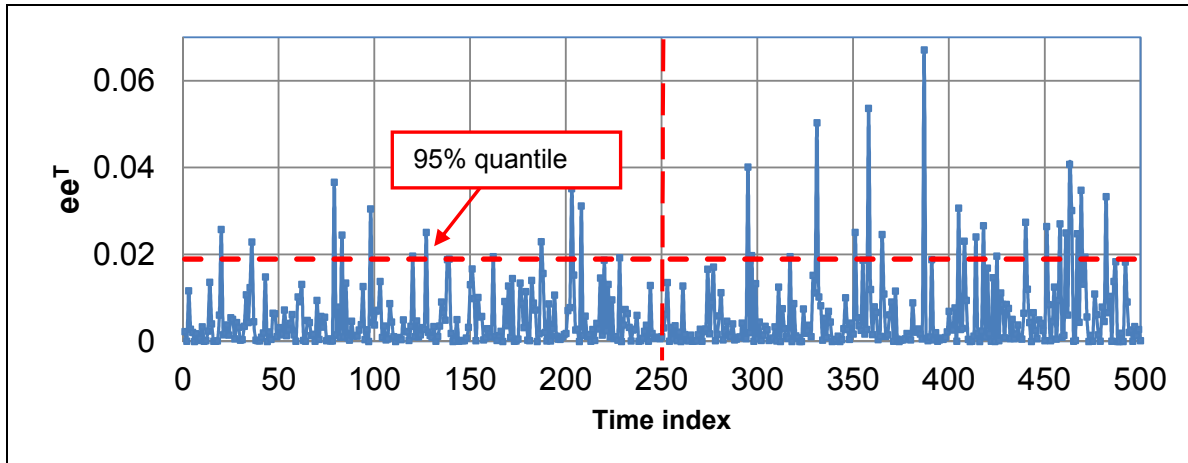


Figure 5.15: Change-point detection with ANN for an autoregressive process of the form $x(t+1) = f(x(t)) + e(0,0.1)$, where e has a zero mean Gaussian distribution with a variance of 0.1. There is a step change in f from 0.9 to 0.5.

5.2.3 Change-Point Detection Using Linear Singular Spectrum Analysis

Figure 5.16: shows the plot of change-point detection using LSSA. LSSA failed to pick up the change-point of the autoregressive system.

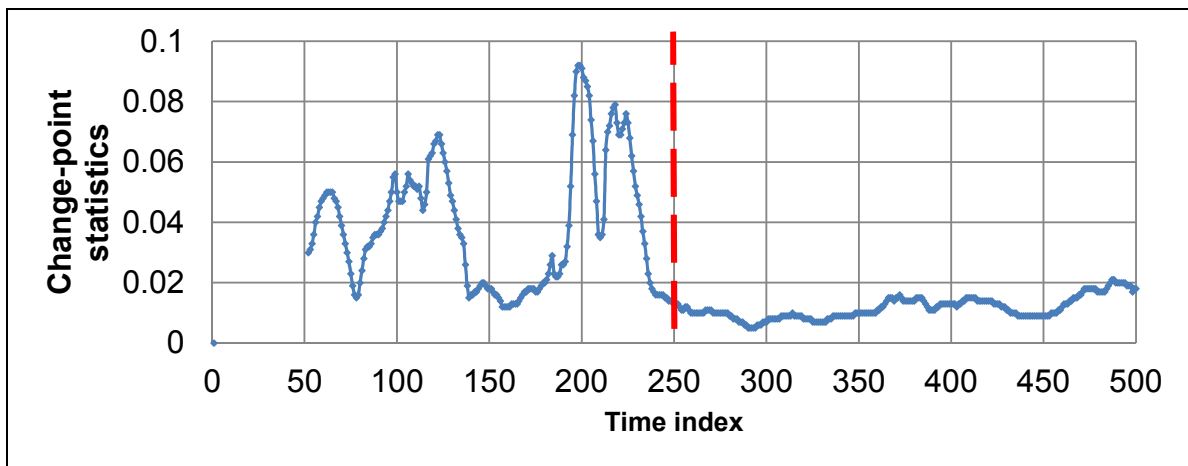


Figure 5.16: Change-point detection with SSA for an autoregressive process of the form $x(t+1) = f(x(t)) + e(0,0.1)$, where e has a zero mean Gaussian distribution with a variance of 0.1. There is a step change a step change in f from 0.9 to 0.5.

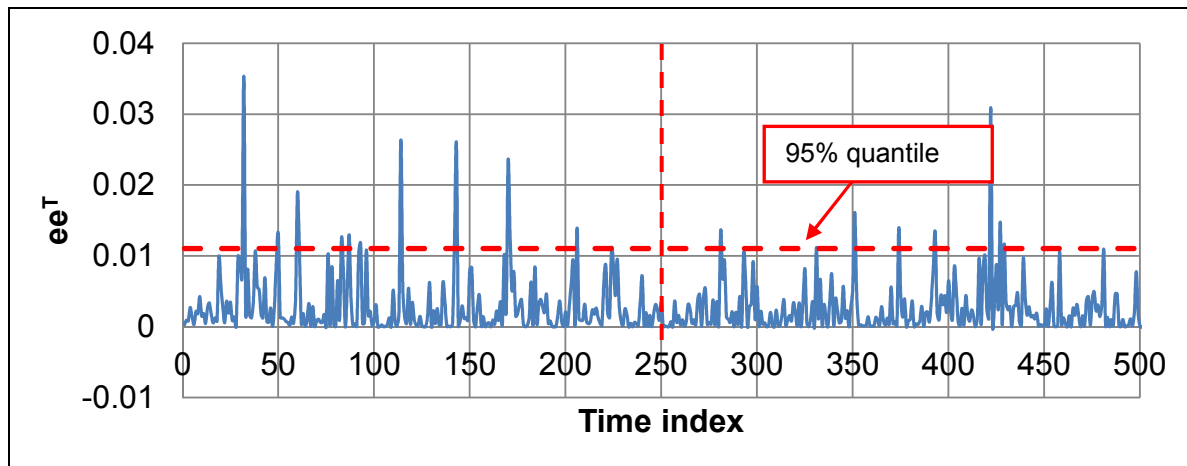


Figure 5.17 : Change-point detection with ANN for an autoregressive process of the form $x(t+1) = f(x(t)) + e(0,0.1)$, where e has a zero mean Gaussian distribution with a variance of 0.1. There is a step change in f from 0.5 to 0.9.

5.2.4 Comparison between NLSSA and LSSA Using FEV

Table 5.4 and Figure 5.18 both show the comparison of NLSSA and LSSA in modelling the time series generated by the autoregressive model. The performances of both NLSSA and LSSA are comparable (similar). The first eigen values in LSSA explains 69% of the variance compared to 65% explained by NLSSA. Figure 5.18 shows the differences between the two models.

Table 5-4: Comparison of nonlinear singular spectrum analysis with linear singular spectrum analysis using explained variance.

Eigennumber	FEV (NLSSA)	FEV (LSSA)
1	0.65	0.69
2	0.79	0.86
3	0.91	0.93
4	1	0.97
5		1

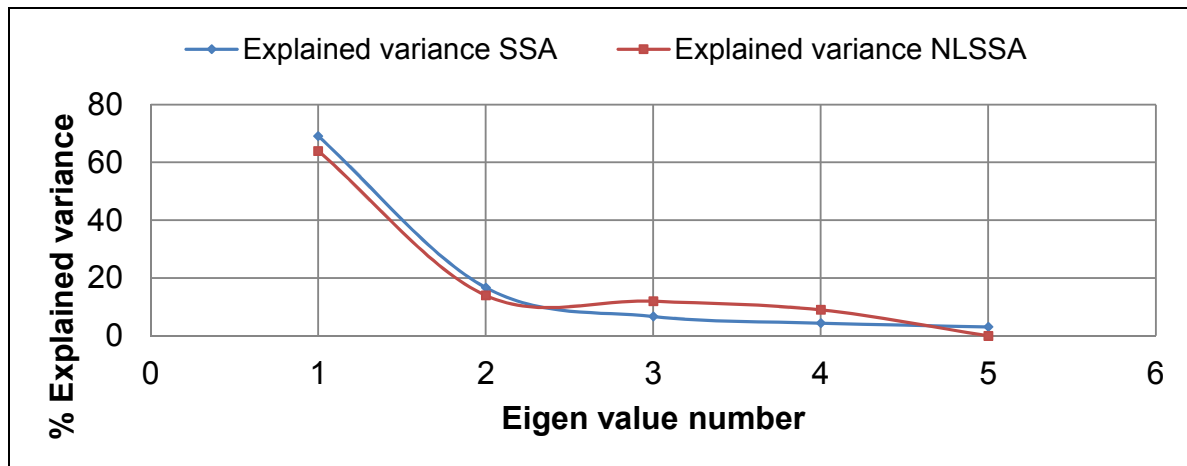


Figure 5.18: The performance of NLSSA and LSSA in modelling the time series.

The performance of most change-point detection methods deteriorate when the data are serially correlated since they implicitly assume that the data are identically and independently distributed (IID) (Hwarnek, 2003; Lund et al, 2007);

5.3 Cross-correlated Time Series (T4)

Two cross-correlated time series of 500 samples each. The first 250 time series samples have a covariance matrix of $C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, while the last 250 samples have a covariance matrix of $C = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$. This simulates an abrupt shift in the correlation between the variables (Figure 5.19).

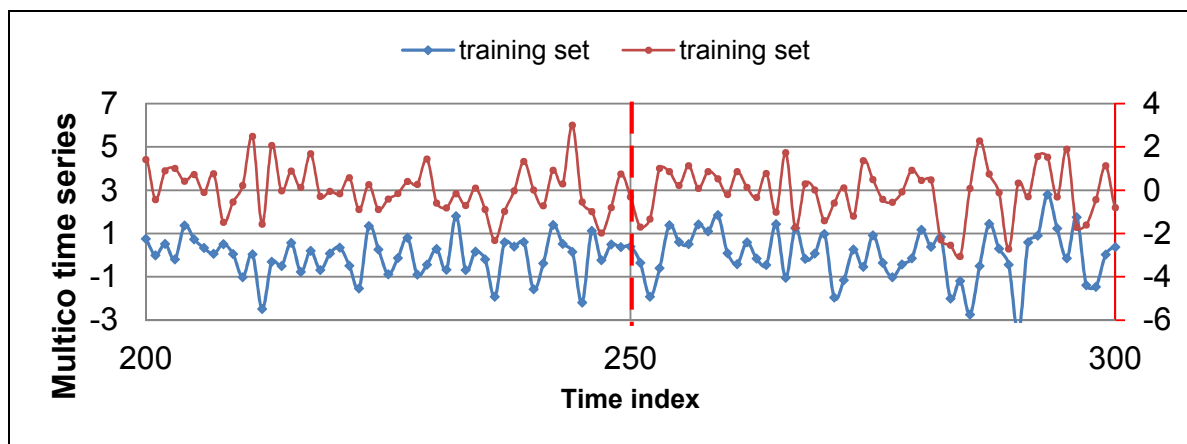


Figure 5.19: Two cross-correlated time series of 500 samples each

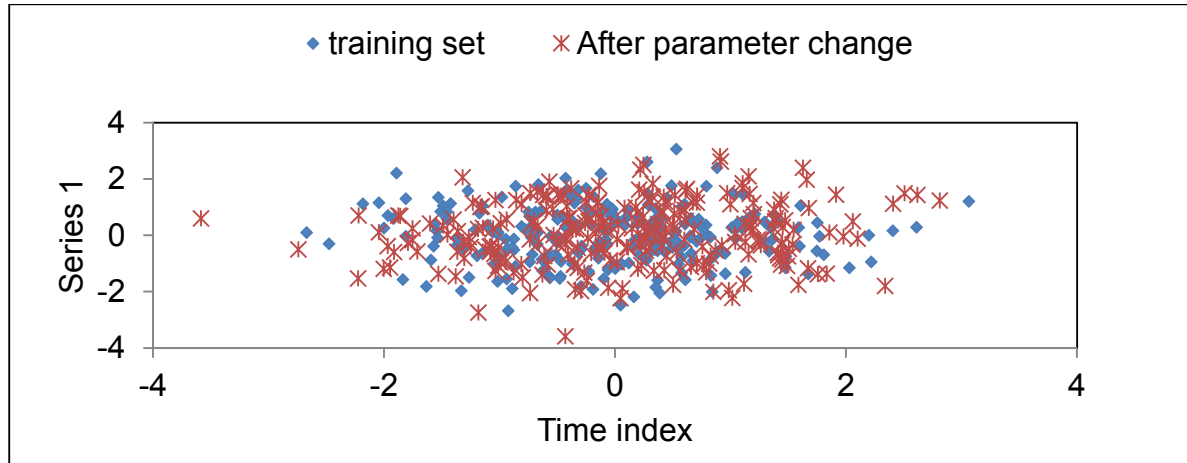


Figure 5.20: Multicorrelation series 1 scatter plot.

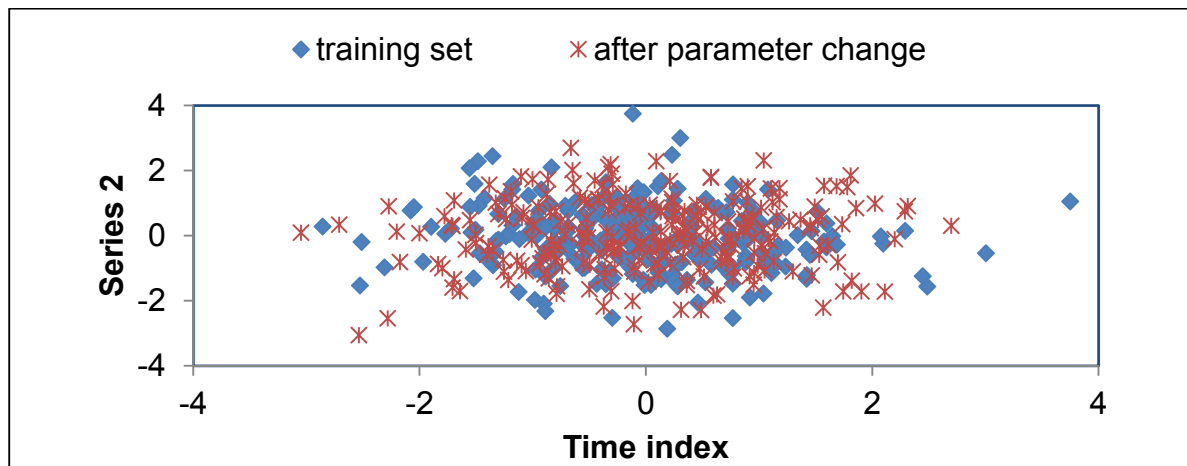


Figure 5.21 : Multicorrelation series 2 scatter plot.

Figures 5.20 and 5.21 show the scatter plots of the multicorrelation time series. From the scatter plots, there is no difference between the training set and the set with a parameter shift, they fall on top of each other, which makes change-point detection difficult.

5.3.1 Network Training

Multicorrelation time series were embedded using an embedding dimension of 5 and time lag of 1. An auto-associative neural network was trained as described in the above sections.

5.3.2 Change-Point Detection with NLSSA

Residual space was developed by simulating data using the trained network and generating residuals. The residual space was analyzed. In Figures 5.22 and 5.23 are

the plots of the squared residuals. There was no change-point detected, as seen from the two graphs.

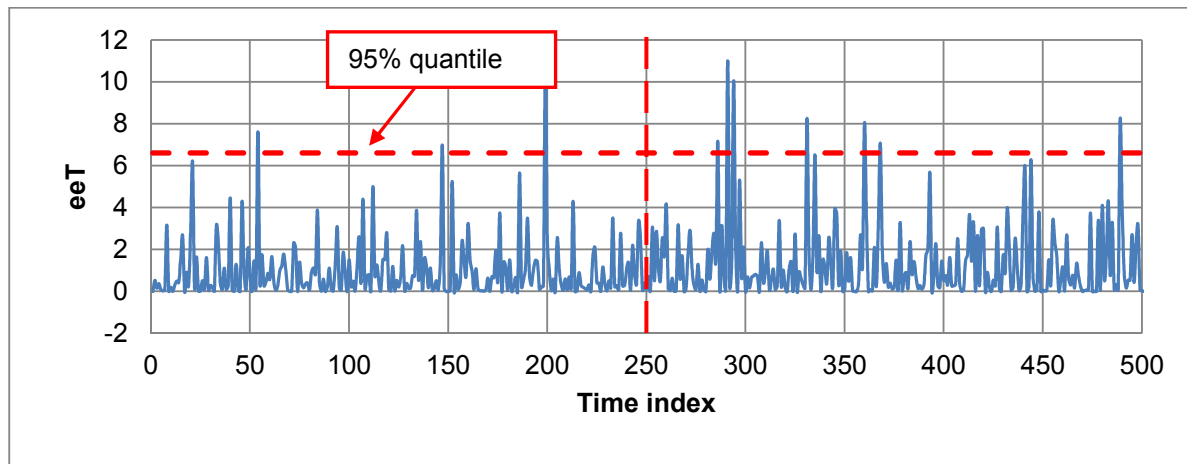


Figure 5.22: Change-point detection for series 1.

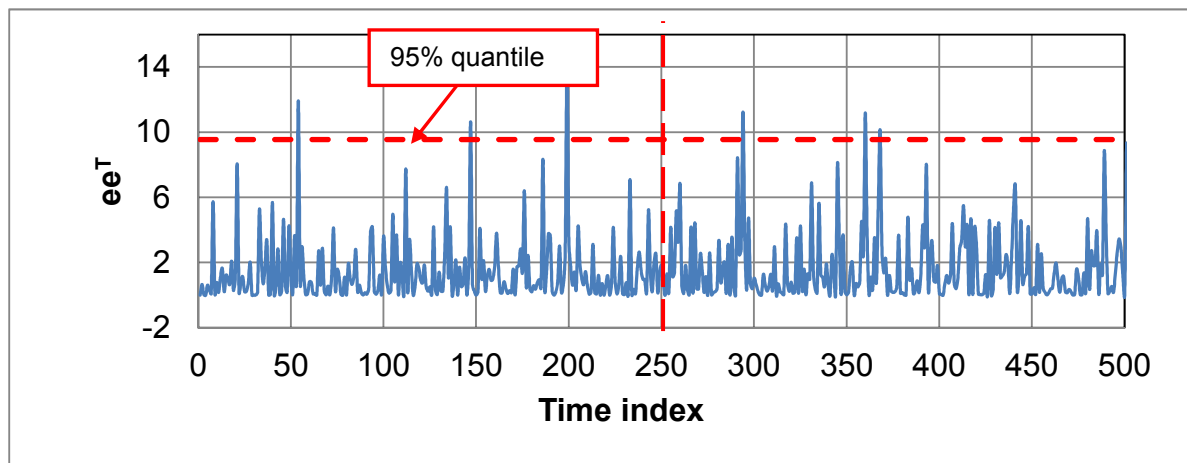


Figure 5.23: Change-point detection for series 2.

Moskvina & Zhigljavsky, 2003 suggested that adding the two correlated time series improves the difference in variance of the two batches before and after the parameter change. Figure 5.24 shows the added time series, as well as the plot of the time series after being simulated using the trained network. Both Figures 5.24 and 5.25 show an improved difference in variance between the series plots before and after a parameter shift, compared to Figures 5.22 and 5.23.

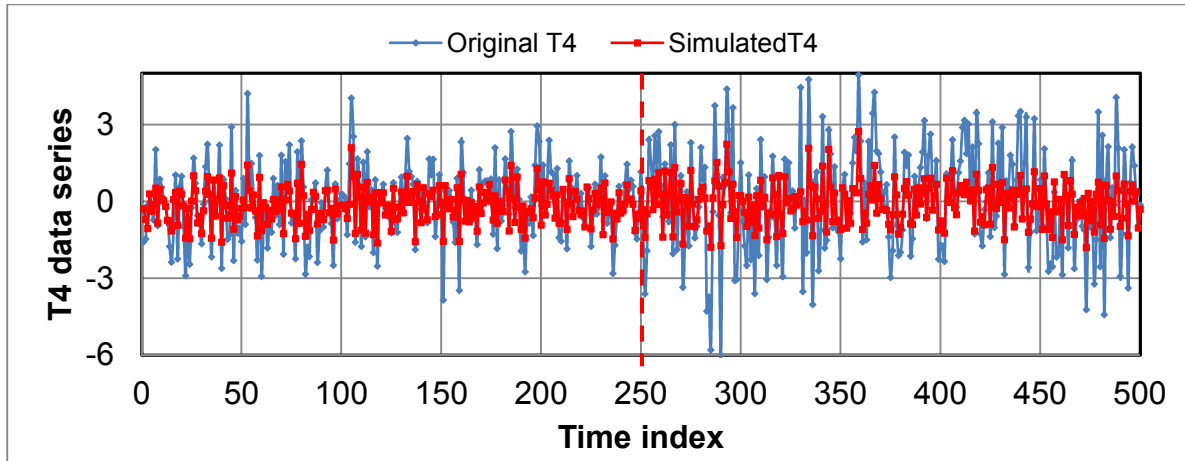


Figure 5.24 : Added time series for a cross-correlated time series.

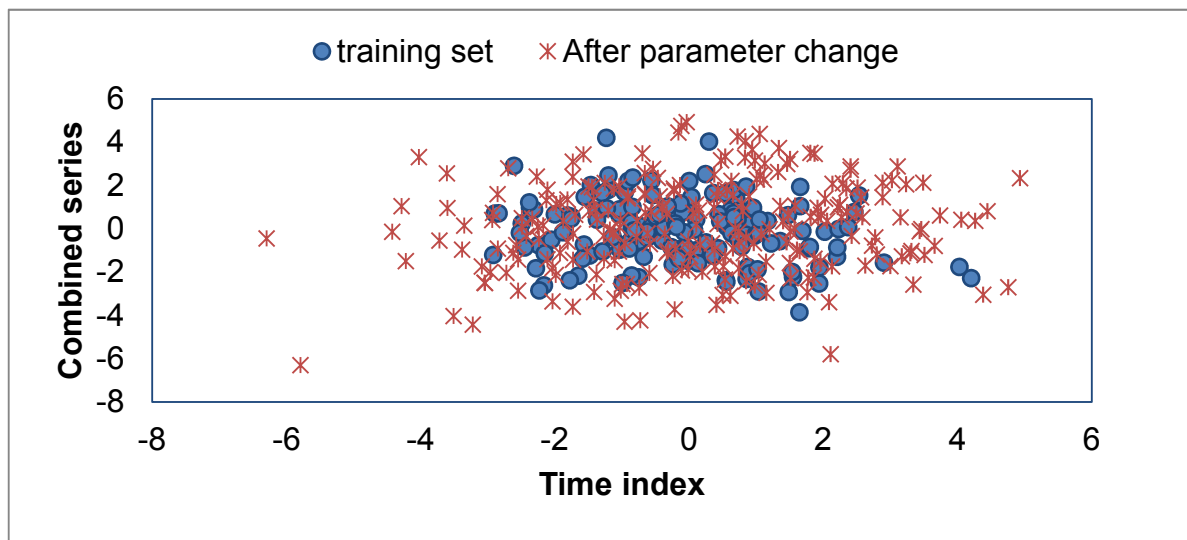


Figure 5.25: Scatter plot of added time series for a cross-correlated time series

5.3.3 Network Training for Multicorrelated Time Series

The resultant time series was embedded using an embedding dimension of 5 and a time lag of 1. Ensembles of networks were trained and the one with smallest MSE and largest regression was selected as the model. Table 5.5 below shows the behaviour of the network with an increase and decrease of neurons. The network behaved the same as the ones trained for T1, T2 and T3. There was no improvement in the performance with an increase in mapping and de-mapping neurons, and an improvement in network performance was observed with an increase in the number of bottleneck neurons.

Table 5-5: Comparing network performances with an increase and decrease of neurons of the mapping, de-mapping and bottleneck layers.

Network structure			Error	Regression			FEV
Mapping layer	Bottleneck layer	De-mapping layer		Training	Validation	Testing	
3	1	3	1.27	0.59	0.52	0.4	0.13
4	1	4	1.25	0.62	0.55	0.46	0.16
5	1	5	1.38	0.59	0.44	0.46	0.14
3	2	3	0.96	0.73	0.68	0.59	0.27
4	2	4	0.87	0.75	0.60	0.61	0.27
5	2	5	0.86	0.77	0.57	0.62	0.26
5	3	5	0.57	0.87	0.78	0.67	0.50
5	4	5	0.30	0.93	0.89	0.92	0.71
5	5	5	0	1	1	1	1

5.3.4 Change-Point Detection with NLSSA

Figure 5.26 shows that a parameter shift took place at time index 250.

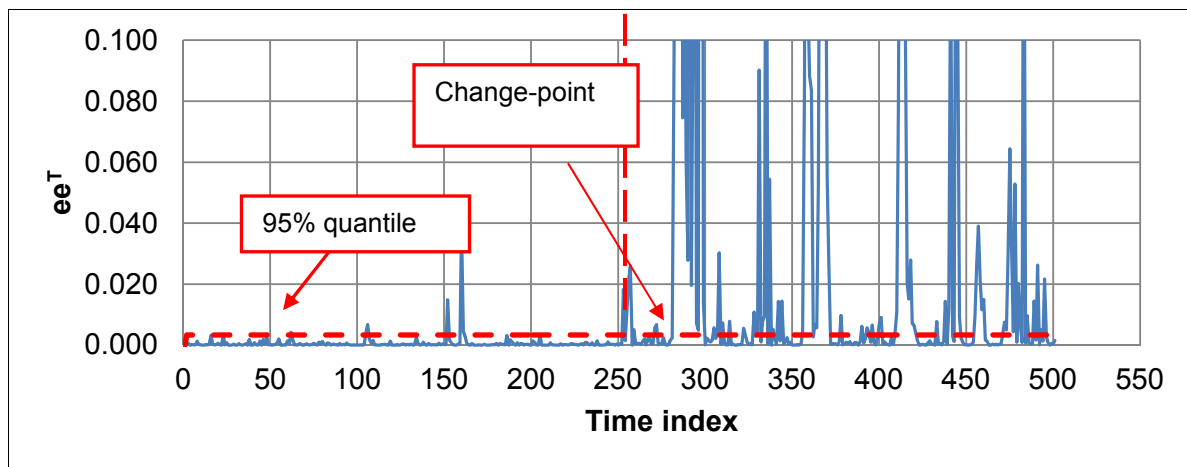


Figure 5.26 : Change-point detection for added cross correlated time series using NLSSA.

5.3.5 Change-Point Detection with LSSA

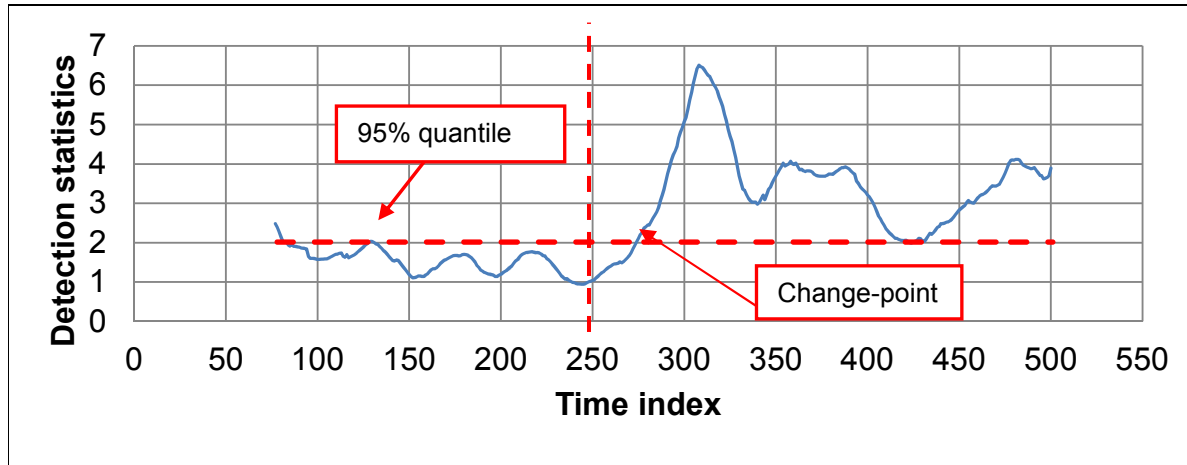


Figure 5.27: Change-point detection for added multicorrelated time series with LSSA

Adding the multicorrelated time series brings out the parameter change. After the time series were added, both NLSSA and LSSA were able to detect the change-point at time index 250 and 275 respectively (Figure 5.26 and Figure 5.27).

5.4 Belousov-Zhabotinsky (BZ) reactions

The Belousov-Zhabotinsky (BZ) reaction is an unstable chemical reaction that maintains self-oscillations and propagating waves, which may display chaos under certain conditions. The BZ reaction equations are shown in Chapter 4; they were solved using MATLAB 7.8.0 (R2009) ODE 45.

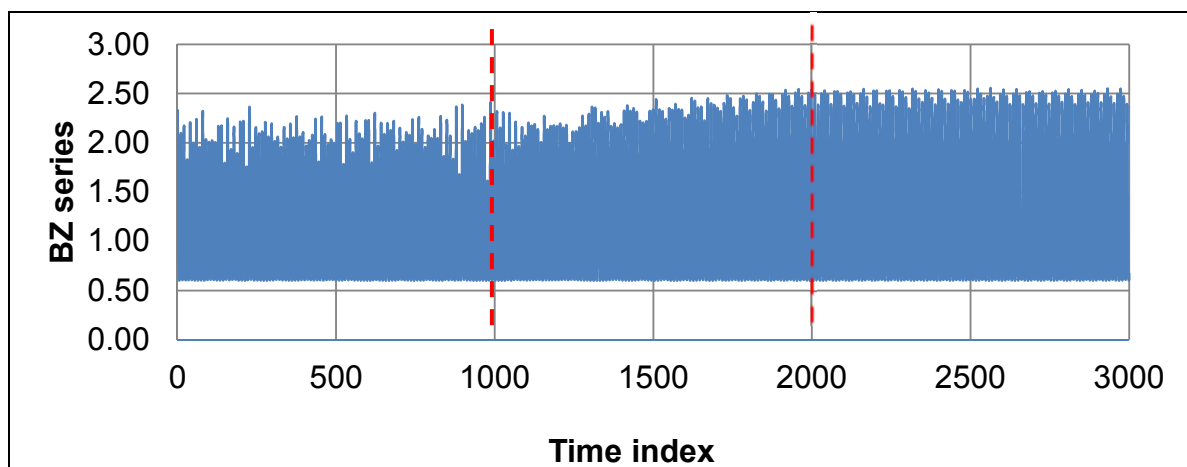


Figure 5.28: Time series for the Belousov-Zhabotinsky (BZ) reaction, with an increase from time index 1000 to 2000.

5.4.1 Calculation of Embedding Parameters

The time series was embedded using an embedding dimension of 5 and a time delay of 2. Average mutual information was used to calculate the time delay, and false nearest neighbour was used to calculate the embedding dimension. Figure 5.29 shows a plot of fraction of false nearest neighbour against embedding dimension. The optimal embedding dimension is the first point where the false nearest neighbour reaches zero

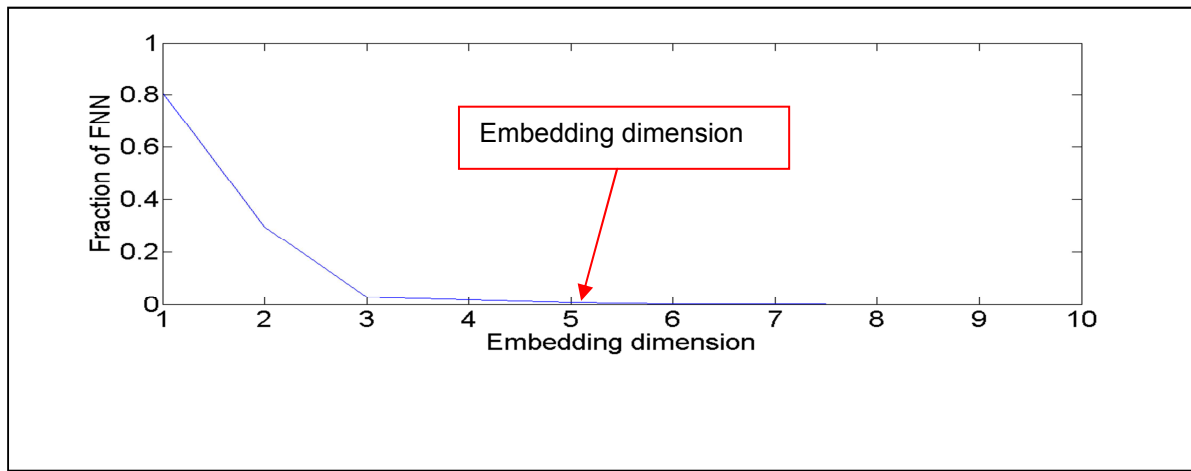


Figure 5.29: Plot of fraction of nearest neighbour and the embedding dimension.

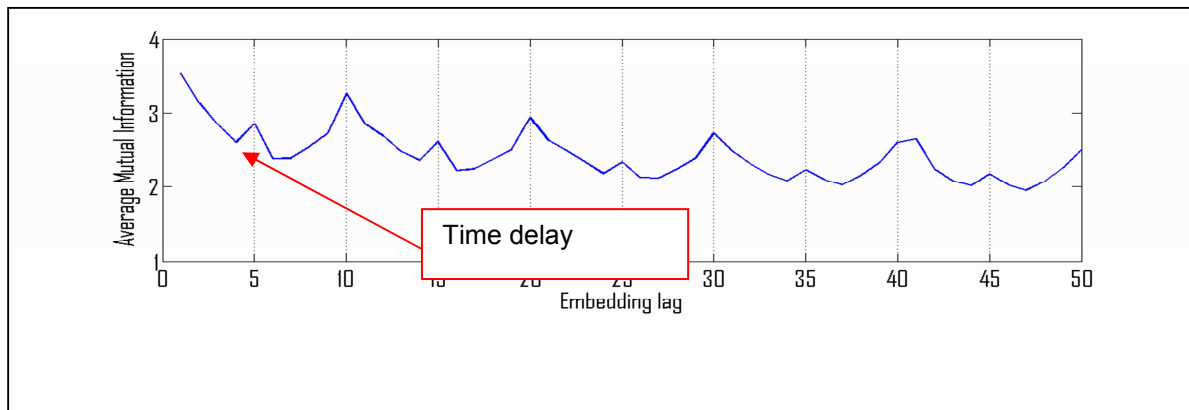


Figure 5.30: Plot of average mutual information vs time delay

Figure 5.30 shows a plot of average mutual information versus embedding lag. The optimum time delay is the first minimum point of the plot of AMI against time lag. Equation B2.1 in the appendix was used to calculate the time delay (average mutual information) and equation C1.7 in the appendix was used to calculate the optimal embedding dimension.

The optimal time delay and embedding dimension should be able to unlock the attractor, Figure 5.31 shows the attractor drawn from the BZ reaction.

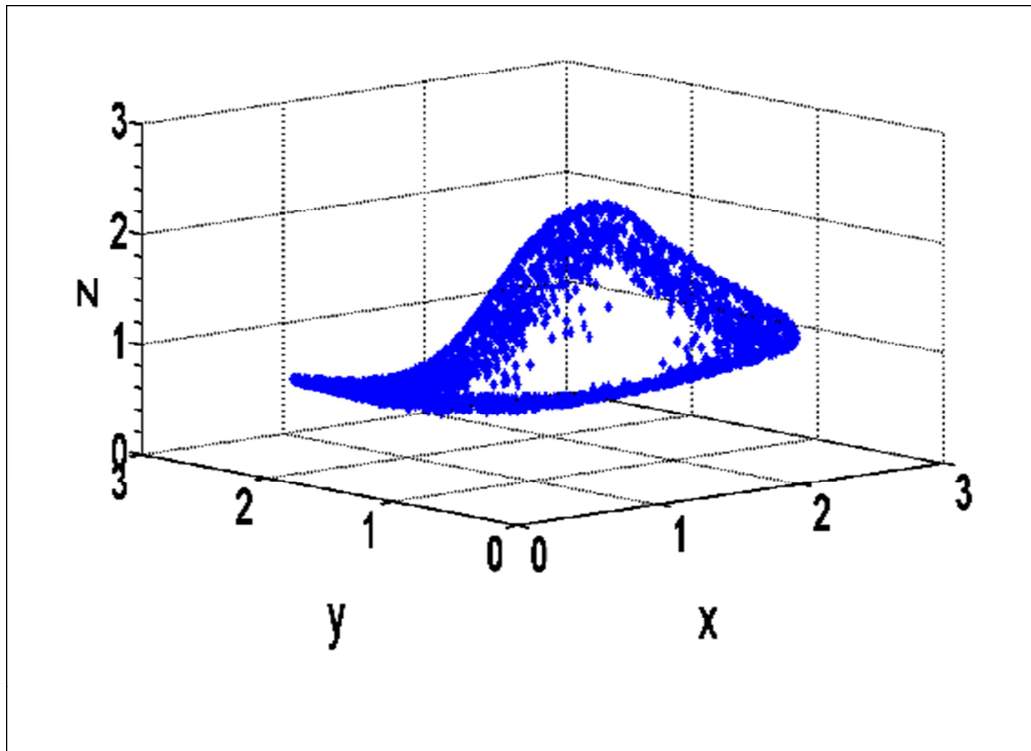


Figure 5.31: Attractor drawn from the BZ system.

5.4.2 Network Training

The first five hundred data points of the BZ data in the control region were used for training the network. A couple of networks were trained and the best one in each configuration was chosen. Table 5.6 shows networks with different configurations and their performances.

Figures 5.32 and 5.33 shows a plot of the BZ time series together with the reconstructed time series using the trained network. Figure 5.32 shows the time series before the change-point and Figure 5.33 shows the time series after the change-point.

Table 5-6: Network performances for the BZ system.

Network structure			Error	Regression			FEV
Mapping layer	Bottleneck layer	De-mapping layer		Training	Validation	Testing	
3	1	3	0.033	0.93	0.91	0.91	0.79
4	1	4	0.027	0.95	0.91	0.94	0.81
5	1	5	0.0203	0.96	0.94	0.95	0.83
6	1	6	0.0089	0.98	0.98	0.94	0.93
7	1	7	0.0069	0.99	0.94	0.99	0.94
3	2	3	0.0172	0.97	0.96	0.96	0.89
4	2	4	0.0041	0.99	0.99	0.99	0.97
5	2	5	0.0033	0.99	0.99	0.99	0.97
6	2	6	0.0027	0.99	0.99	0.99	0.98
7	2	7	0.0024	1	0.99	0.99	0.98

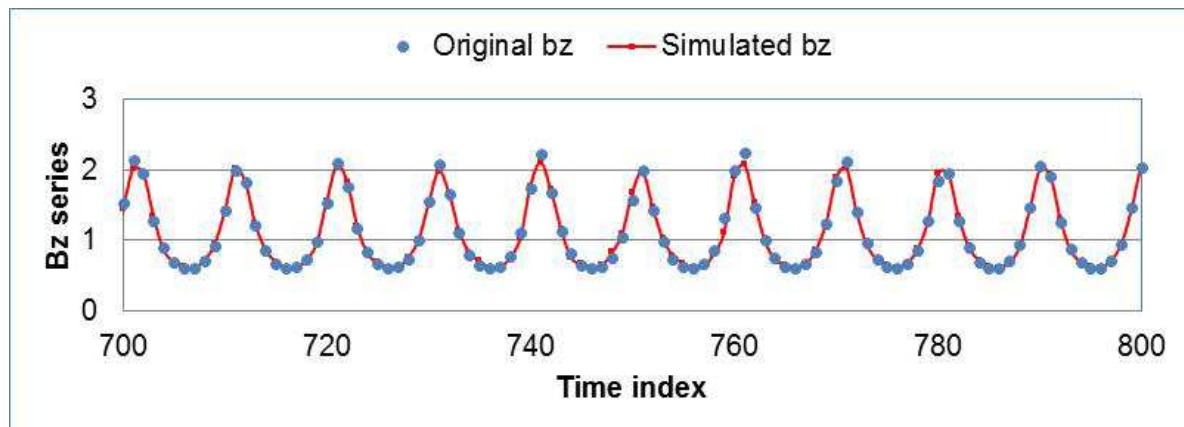


Figure 5.32: Plot of the Belousov-Zhabotinsky (BZ) time series, together with the same series simulated by the model before the change-point.

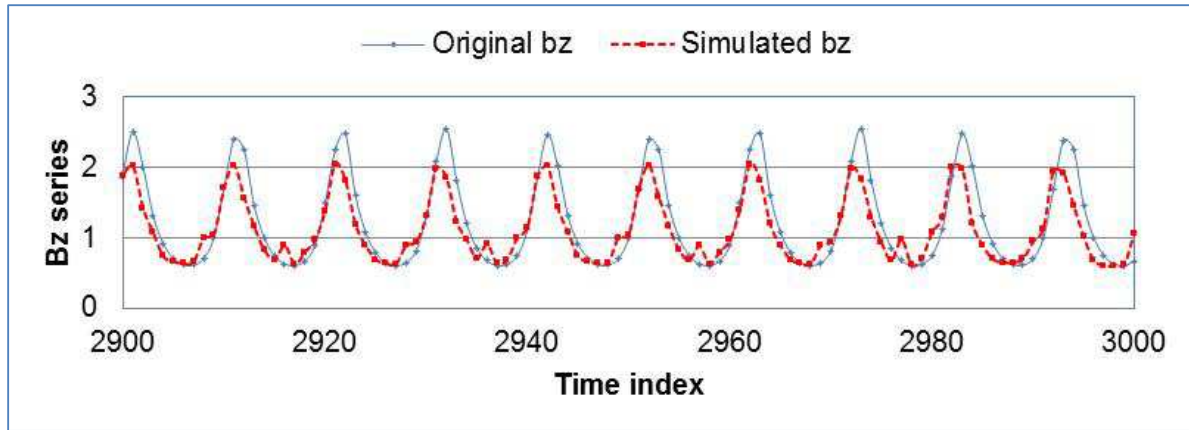


Figure 5.33: Plot of the Belousov-Zhabotinsky (BZ) time series, together with the same series simulated by the model after the change-point.

Figures 5.4 a) and 5.34 b) show the behaviour of the network when increasing the number of neurons. The network performance improves with an increase in the number of neurons. With the previous data, the network performance did not improve with an increase in the mapping or de-mapping neurons.

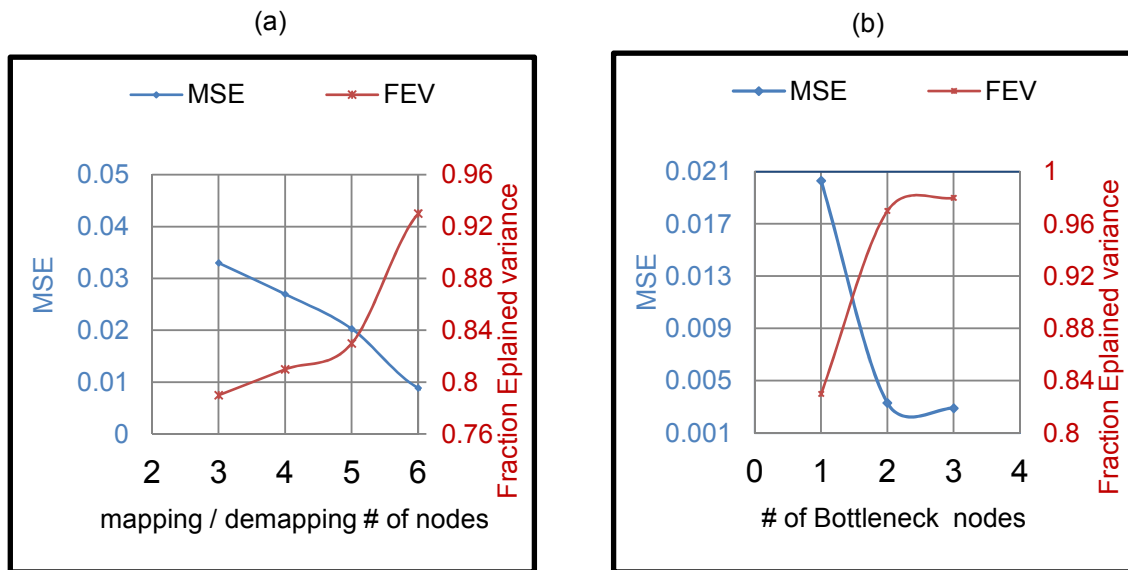


Figure 5.34 a) Plot of the network behaviour when increasing the number of neurons of the mapping and de-mapping layers, b) Plot of the network behaviour when increasing the number of bottleneck neurons.

The FEV increased from 0.79 in a network with three neurons in the mapping / de-mapping layers to 0.94 for a network with 7 neurons for the mapping / de-mapping layers. All networks have one node for the bottleneck. For the same networks, the MSE moved from 0.03 to 0.007. The improvement is true until the number of

neurons reach a point where they start to over-fit the data series. The plot of bottleneck number of neurons against MSE and FEV shows an improvement in the network performance with an increase in the number of bottleneck neurons. A network with 5 neurons for the mapping / de-mapping layers had its MSE move from 0.02 to 0.003 with an addition of only one node on the bottleneck and the variance increased from 0.83 to 0.97.

5.4.3 Change-Point Detection with NLSSA

The trained network was used to generate residuals: $e_i = x_i - \hat{x}_i$, where x_i is the expected value and \hat{x}_i is the simulated value. The residuals were squared, that is, $e_i e_i^T = (x_i - \hat{x}_i)^2$. A 95% quantile was calculated for the part before the parameter shift. Points above the 95% quantile have a high probability of being the result of a parameter shift. To minimise on false alarms, a 99% quantile can be used. This will reduce false alarms. Figure 5.35 shows the change-point detection of the BZ reactions. The graph shows some false alarms before the actual parameter shift at time index 1000. The parameter shift is only detected at time index 1200, this is probably because it is a ramp rather than abrupt shift.

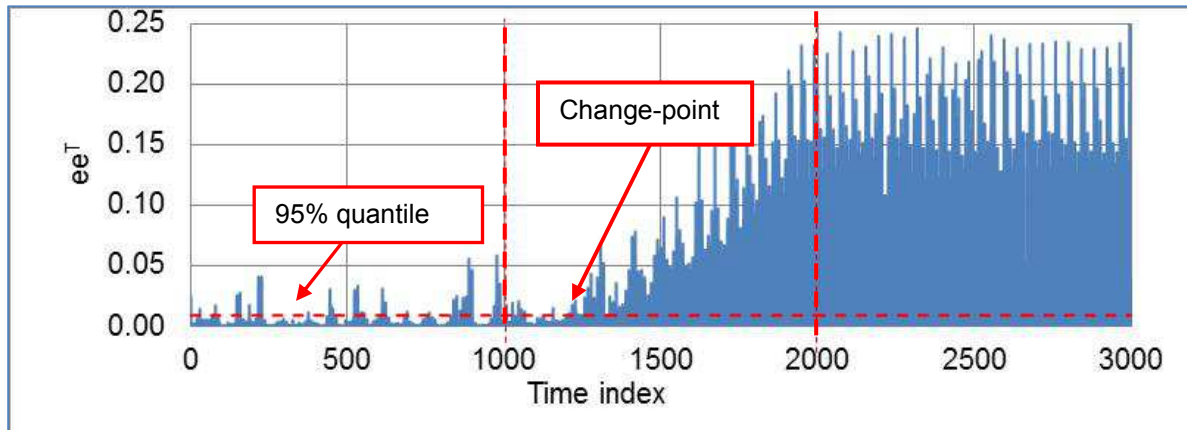


Figure 5.35 : Change-point detection of BZ reactions by use of NLSSA.

5.4.4 Change-Point Detection with LSSA

Linear singular spectrum analysis was able to pick up the change-point at time index 1200 in Figure 5.36. It can be observed that the delay in detecting the change was caused by the fact that the change is a slow ramp rather than an abrupt shift.

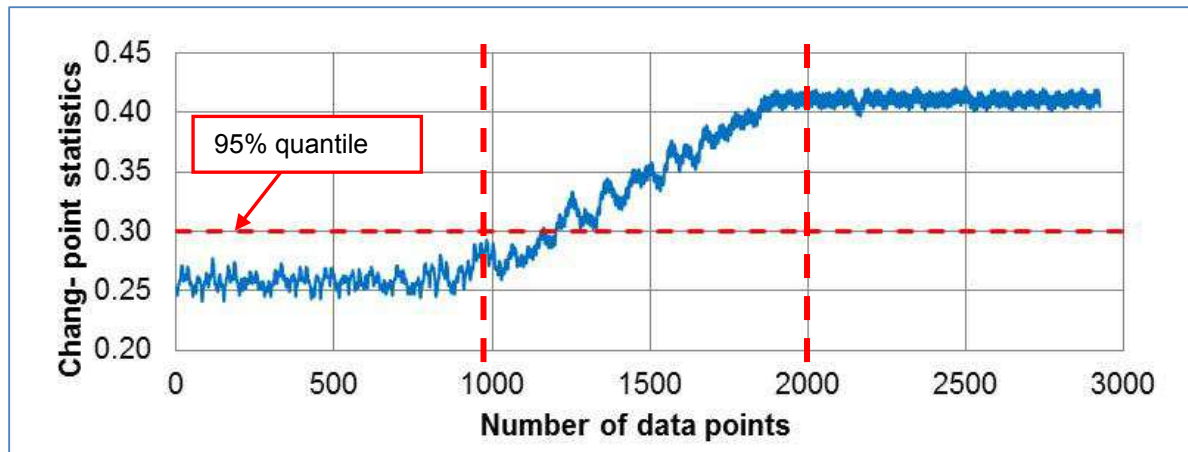


Figure 5.36: Change-point detection with LSSA for a BZ time series

5.4.5 Comparison between NLSSA and LSSA

Table.5-7: FEV for NLSSA and LSSA

Eigennumber	FEV (NLSSA)	FEV (LSSA)
1	0.94	0.88
2	0.98	0.97
3	1	1

The first eigennumber for nonlinear singular spectrum analysis explains 94% of the variance of the BZ reactions, whereas the linear singular spectrum analysis explains 88%. Two eigennumbers in NLSSA explains 98% whilst two eigennumbers in LSSA explains 97%. Both methods perform well for the BZ reactions, although the NLSSA seems to be superior.

5.4.6 Feature Extraction with NLSSA

Feature extraction refers to identifying the salient aspects or properties of data. The bottleneck in NLSSA compresses the data to a lower dimension and in that process, captures the important aspects of the data. The number of neurons in the bottleneck layer determines the order of data reduction. There are two ways of extracting features from the NLSSA (Kerschen and Golinval, 2003). The first one is called modal analysis: the neural network has one node for the bottleneck. The data extracted from the bottleneck will be the first principal component (pc1). Residuals generated by the first network will be used as an input to the second network. Data extracted from the bottleneck of the second network will be the second principal component (pc2). Residuals from the second network will be the input for the third

network and data extracted from the bottleneck becomes the third principal component (pc3). The residual of the i^{th} network is the input of the $(i + 1)^{\text{th}}$ network

The second method of extracting features from NLSSA is called non-modal analysis. In non-modal analysis, the number of bottleneck neurons is set to be equal to the number of wanted features. This second method is more reliable since networks with more than one neuron for their bottleneck are more stable and result in a higher fraction of explained variance compared to the one with one neuron.

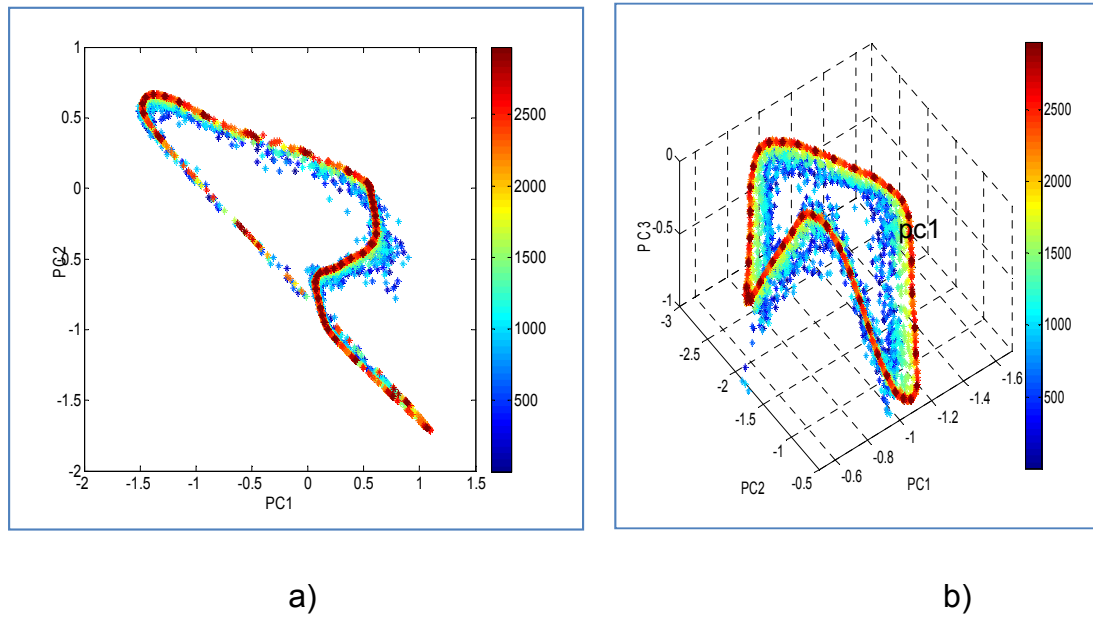


Figure 5.37: a) Plot of pc1 and pc2 obtained by training a network with two neurons for the bottleneck, b) Plot of the features extracted from a network with three neurons for the bottleneck

Figure 5.37 a) shows a plot of pc1 and pc2, where pc1 and pc2 were extracted from the bottleneck of an auto-associative neural network with two neurons for its bottleneck. Figure 5.37 b) shows a three dimensional plot of pc's obtained from training a network with three neurons for the bottleneck. The colours in Figures 5.37 a) and 5.37 b) represent the history of the time series. The blue to greenish colour represent the first 1000 points with $C_f = 4.5 \times 10^{-4} \text{ s}^{-1}$.

The greenish to yellowish colour represents the points from 1000 to 2000, where the series ramp from $C_f = 4.5 \times 10^{-4} \text{ s}^{-1}$ to $C_f = 5.0 \times 10^{-4} \text{ s}^{-1}$. The red to dark red colour represents the series points from 2000 to 3000 points where $C_f = 5.0 \times 10^{-4} \text{ s}^{-1}$. The fact that the colours in Figures 5.40 and 5.41 show distinct regions mean that there

has been a parameter change. If the parameter change was not there, the colours would be mixed up in one line.

5.5 Autocatalytic Reaction System

A detailed description of the autocatalytic reaction system is found in Chapter 4. The autocatalytic reaction system was solved with ODE45 subroutine in MATLAB 7.2 at a sampling rate of 0.005. For the first 10 000 observations, the same parameters as specified by Lee and Chang (1996) were used. A simulated fault condition was then introduced through a slight increase in the feed concentrations of A and D . This caused an increase in the parameters γ_1 and γ_2 . As with the previous two cases, the parameters γ_1 and γ_2 were allowed to drift slowly from their starting values, $\gamma_1 = 1.5$ and $\gamma_2 = 4.2$, to $\gamma_1 = 1.55$ and $\gamma_2 = 4.25$, after which the parameters were again kept constant at their new values, $\gamma_1 = 1.55$ and $\gamma_2 = 4.25$, for another 10 000 observations. The 30 000 points were then sampled to get a series with 3000 points where the first 1000 points have $\gamma_1 = 1.5$ and $\gamma_2 = 4.2$. The second 1000 points have $\gamma_1 = 1.5$ and $\gamma_2 = 4.2$ ramping up to $\gamma_1 = 1.55$ and $\gamma_2 = 4.25$ and the last 1000 points have $\gamma_1 = 1.55$ and $\gamma_2 = 4.25$ constant. Figure 5.38 below is a plot of the time series.

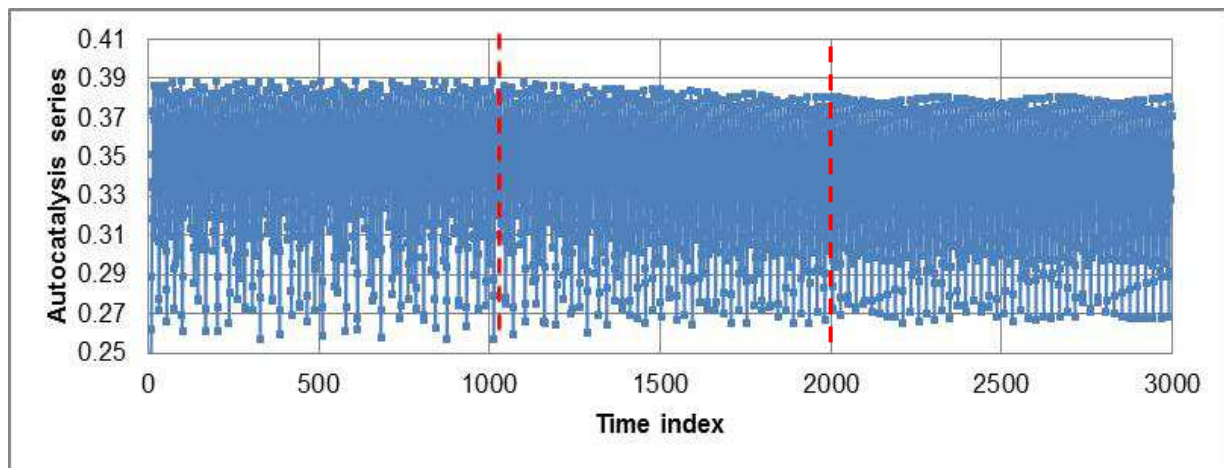


Figure 5.38: Plot of autocatalysis time series.

5.5.1 Calculation of Embedding Parameters

The time series was embedded using an embedding dimension of 5 and a time delay of 2. Average mutual information was used to calculate the time delay and false nearest neighbour was used to calculate embedding dimension. Figure 5.39 shows a plot of average mutual information against time delay. The optimum time delay is the

first minimum point of the plot of AMI against time lag. See equation B2.1 in the appendix for equations used to calculate average mutual information and equation C1.7 in the appendix for equations used to calculate false nearest neighbour.

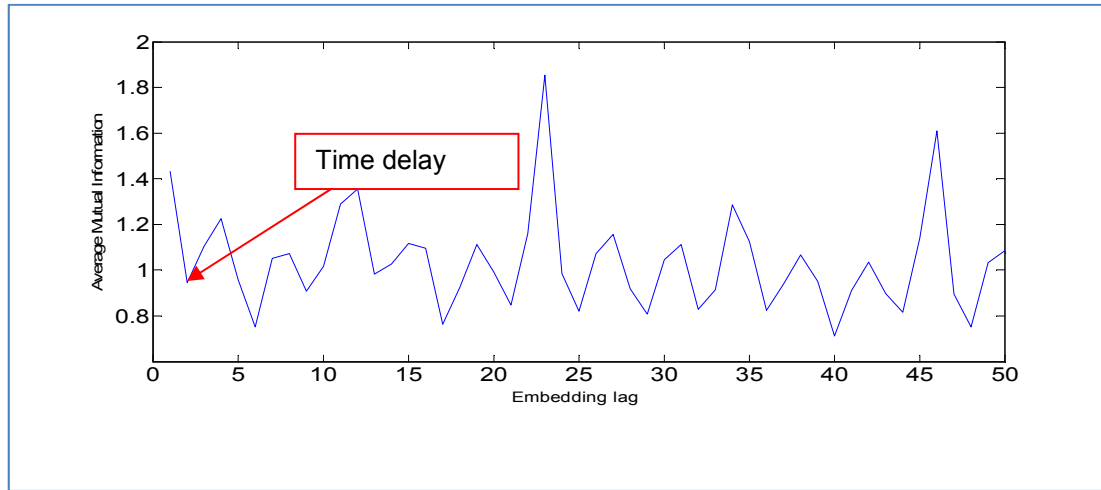


Figure 5.39: Average mutual information vs time delay.

Figure 5.40 shows a plot of fractions of false nearest neighbour against embedding dimension. The optimal embedding dimension is the first point where the false nearest neighbour reaches zero.

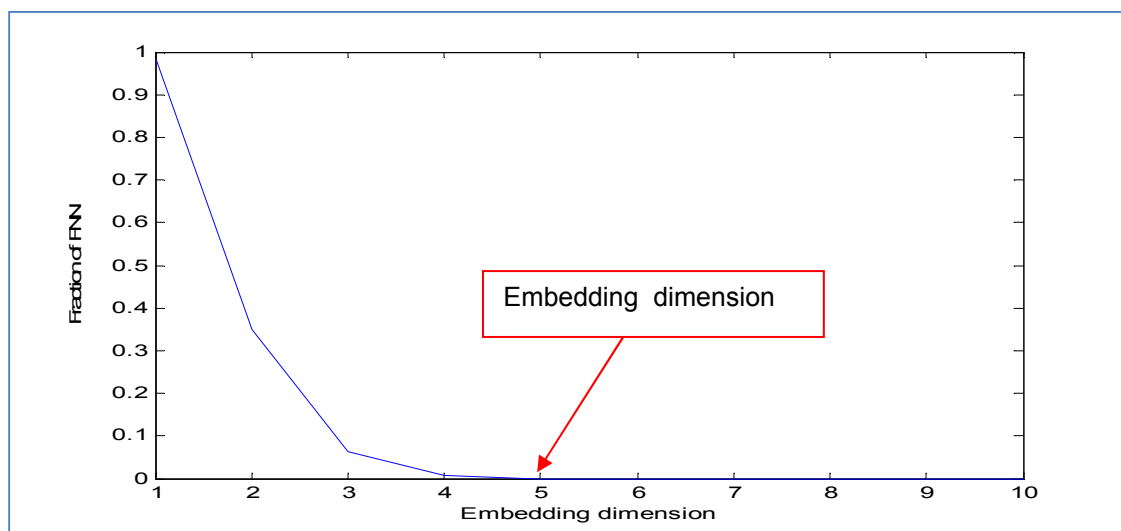


Figure 5.40: Plot of fraction of nearest neighbour against the embedding dimension.

An optimal time delay and embedding dimension should be able to unlock the attractor, Figure 5.41 shows the attractor drawn from the autocatalytic reaction.

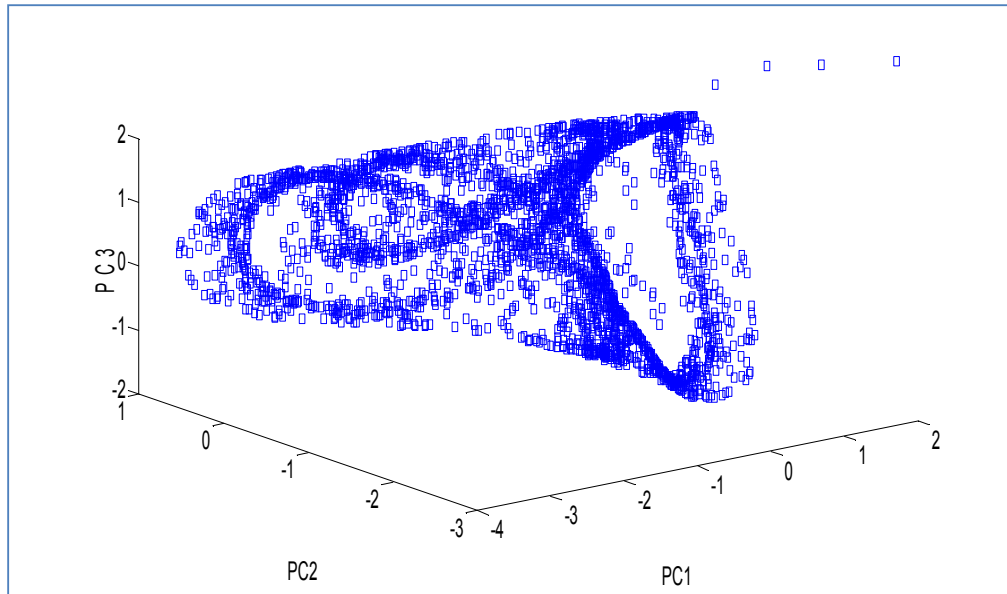


Figure 5.41: The attractor for the autocatalytic reaction.

5.5.2 Network Training

The first five hundred data points of the autocatalytic process were used for training the network. An ensemble of networks was trained and the best one in each configuration was chosen. Table 5.8 shows networks with different configurations and their performances. Figures 5.42 and 5.43 show the autocatalytic time series together with the reconstructed time series using the trained network.

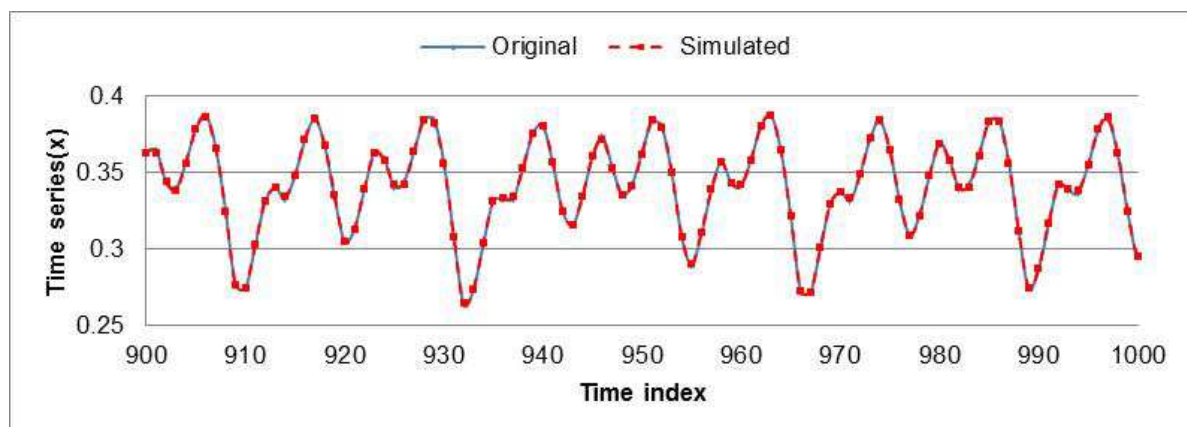


Figure 5.42 : Plot of the first 100 points of the autocatalysis original data series (series x) and its simulated series.

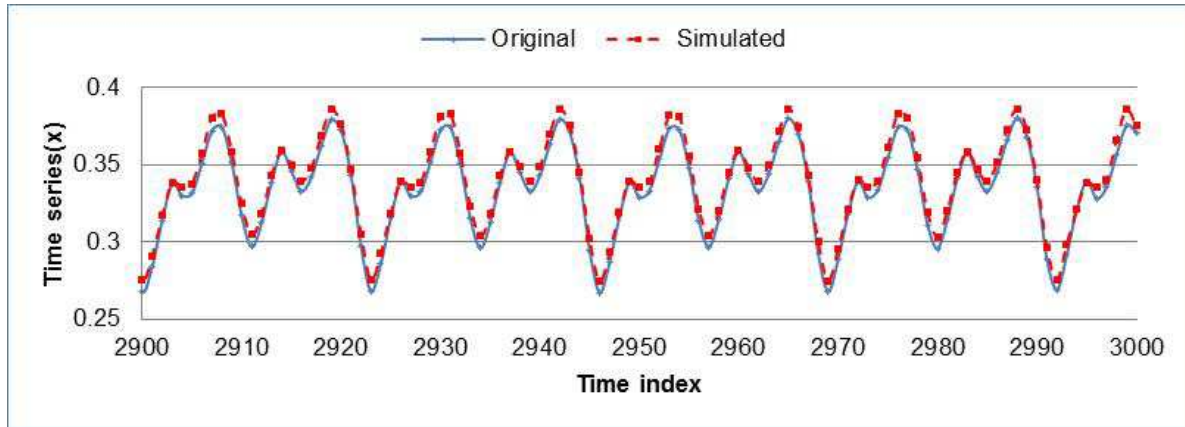


Figure 5.43: Plot of the last 100 points of the autocatalysis original data series (series x) and its simulated series.

Figure 5.42 shows the first 1000 data points of the autocatalysis reaction. The first 1000 points consists of the 'in control region'. That is the reaction before the fault was introduced. It has $\gamma_1 = 1.5$ and $\gamma_2 = 4.2$

The last 1000 points have $\gamma_1 = 1.55$ and $\gamma_2 = 4.25$. The training set is taken from the first 1000 points. It can be seen from Figure 5.42 that the original series and the simulated series are the same. Figure 5.46 shows that the original and the simulated series do not lie on top of each other, as in Figure 5.43. This is a result of the induced disturbances.

Table 5-8 : Networks with different configurations and their performances for the autocatalytic reaction system.

Network structure			MSE error	Regression			FEV
Mapping layer	Bottleneck layer	De-mapping layer		Training	Validation	Testing	
3	1	3	4.71×10^{-4}	0.75	0.68	0.74	0.30
4	1	4	4.01×10^{-4}	0.77	0.79	0.78	0.36
5	1	5	3.65×10^{-4}	0.83	0.76	0.79	0.43
3	2	3	2.62×10^{-4}	0.87	0.89	0.83	0.61
4	2	4	1.26×10^{-4}	0.93	0.94	0.94	0.77
5	2	5	8.43×10^{-5}	0.97	0.96	0.97	0.87
5	3	5	9.94×10^{-6}	1	0.99	0.99	0.98

5.5.3 Change-Point Detection with NLSSA

Figure 5.44 below shows the change-point detection of the autocatalytic reaction. A network with two nodes in the bottleneck and five nodes for the mapping and de-mapping layers was trained and used to generate residuals for this system. Smaller networks failed to detect the change, the mean square error for the network training was 8.43×10^{-5} and the training regression was 0.96 (see Table 5.8). The magnitude of the error is also too small (see y-axes of Figure 5.44). Figure 5.44 shows the start of the ramp at time index 1000 and the increase is nicely visible until time index 2000 where it remains constant for the next 1000 points. A 95% quantile is plotted in the figure to show where the change-point takes place.

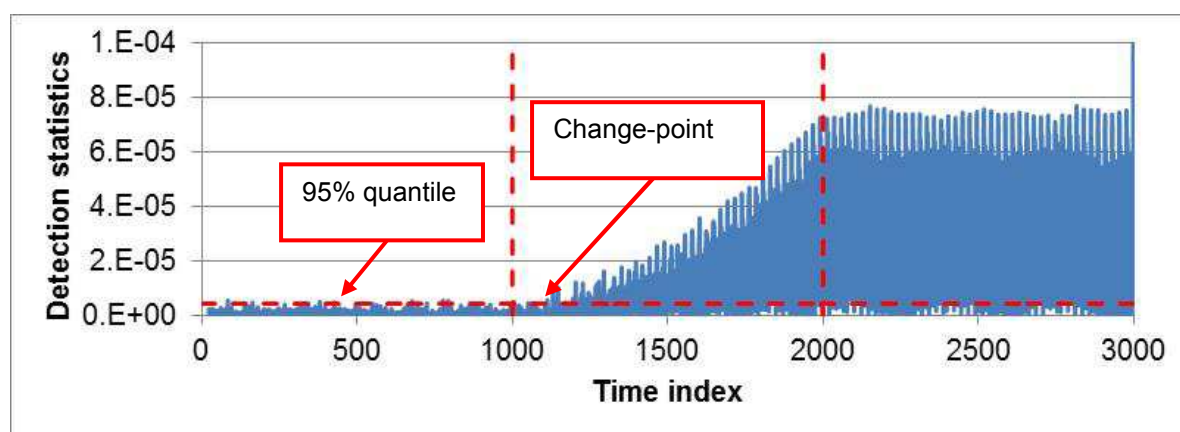


Figure 5.44: Change-point detection for autocatalytic reactions with NLSSA

5.5.4 Change-Point Detection with LSSA

A linear singular spectrum analysis change –point detection algorithm was applied to the autocatalytic time series. The following parameters were used in the algorithm: # change-point parameters: M=25, Center (no), PC: 1-2, N=45, p=22, q=45. Figure 5.45 shows the result obtained from the LSSA algorithm. LSSA fails to detect change. It only shows false change-points on the first 1000 index points of which there are no change-points expected.

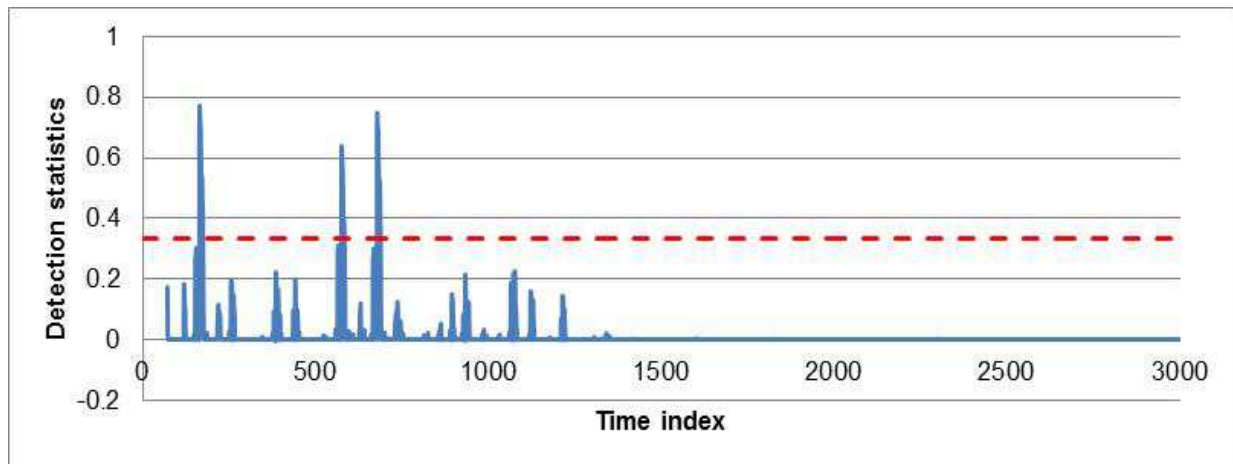


Figure 5.45: Change-point detection for autocatalytic reactions with LSSA.

5.5.5 Comparison between NLSSA and LSSA

Table 5.9 below shows that LSSA is able to capture most of the information in the autocatalytic reactions. Its first eigenvalue captures 99% of the total information whilst NLSSA captures 44%. The second eigenvalue of NLSSA captures 89% and the third eigenvalue captures 99.6%. Since the introduced fault in the autocatalytic system is very small, nonlinear LSSA fails to capture it and thus fails to detect the change. On the other hand, NLSSA is able to capture the change. It is able to capture it properly when the network has two nodes and above in its bottleneck.

Table 5-9: showing FEV for NLSSA and LSSA

Eigennumber	FEV (NLSSA)	FEV (LSSA)
1	0.442	0.995
2	0.890	0.998
3	0.996	1

5.5.6 Feature Extraction with NLSSA

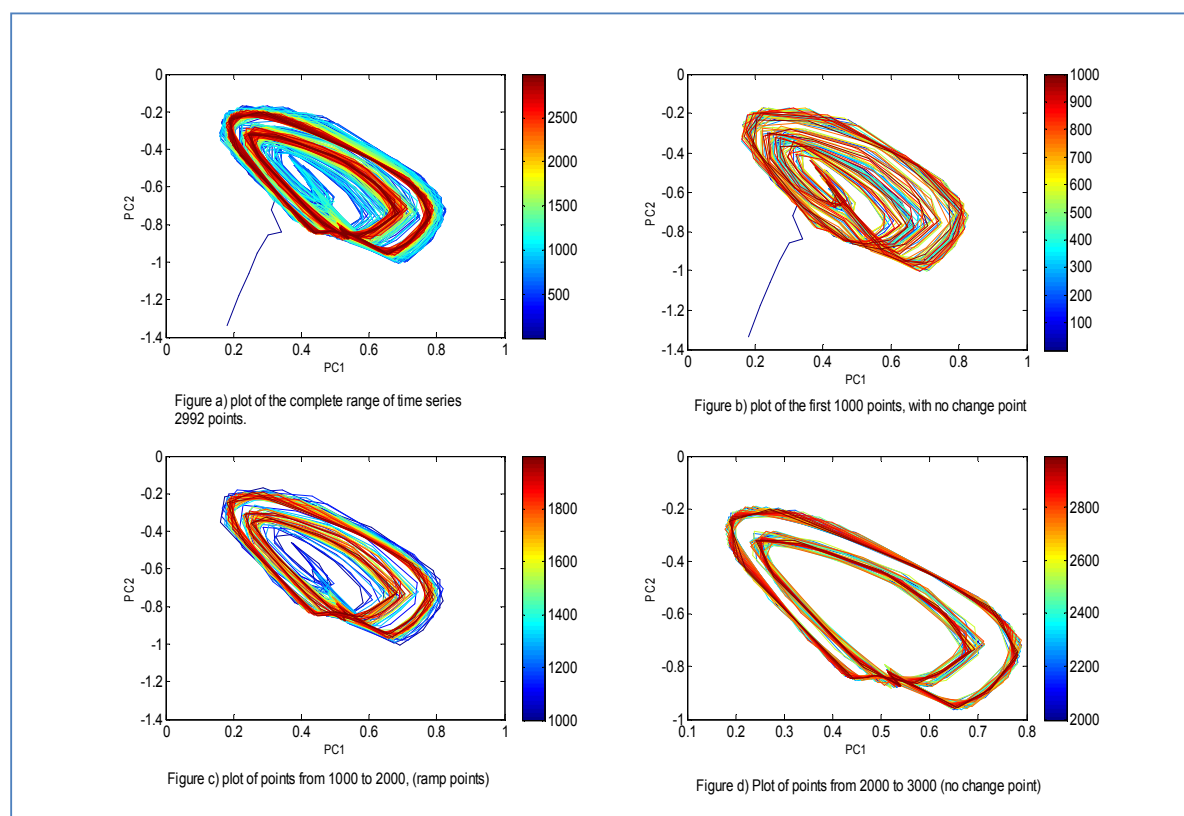


Figure 5.46 : Plot of features from the autocatalytic reactions.

Figure 5.46 a) is the plot of the 3000 points of the sampled autocatalytic reaction system. The colours of the plot show that the time series changes with time. From point 1000 to 2000 the time series increases in magnitude, shown by the greenish to yellowish color. After that, the time series is stable as shown by the reddish colour. Figure 5.46 b) is a plot of the first 1000 points.

The colours in this plot are not separated; this is because there is no increase or decrease in the magnitude of the data. Figure 5.46 c) is a plot of the data series from 1000 to 2000. This is the region where the concentration of reactant A and D is slowly increased to cause a change in the reaction system. It can be seen from the figure that with time, points of the time series change. The colours are not on top of each other. The last plot Figure 5.46 d) is the plot of data points from time index 2000 to 3000. In this segment, the disturbance caused by the increase in concentration of reactants A and D have normalised and are constant, therefore there is no change in the data. The colours of the time series are mixed because the time series does not change with time. Figure 5.46 shows that by analyzing features

one should be able to tell whether there was a change in the system being analyzed or not. The only shortfall is that determining the exact time index where the change occurred is not possible. Figure 5.47 shows a plot of three principal components taken from training a network with three nodes on the bottleneck. The same feature observed in Figure 5.46 is also present in the three dimensional plot, with the colours showing the different states of the time series.

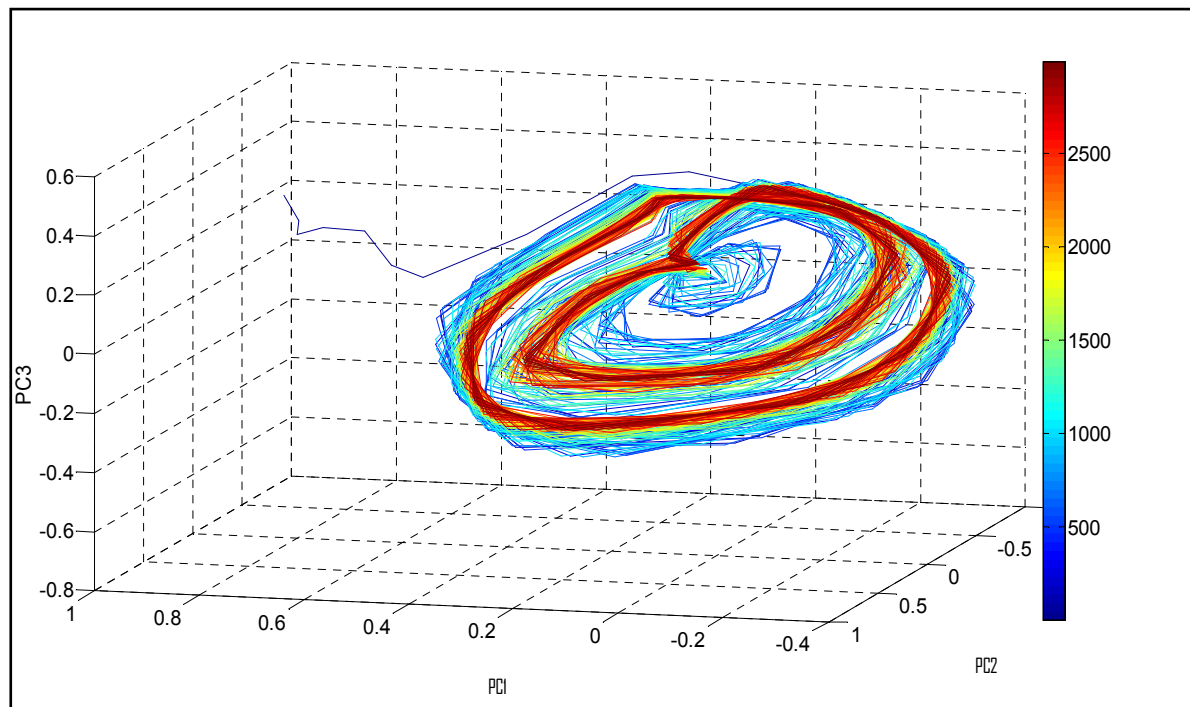


Figure 5.47 : Plot of pc1, pc2, and pc3 obtained from training a network with 3 nodes for the bottleneck.

5.6 Predator-Prey

The predator – prey system is described in Chapter 4. Figures 5.48 and 5.49 show the plot of the predator-prey time series. A parameter shift is introduced at the 10 000th observation and step increments added until the 19 000th observation and thereafter all parameters were kept constant until the 30 000th observation.

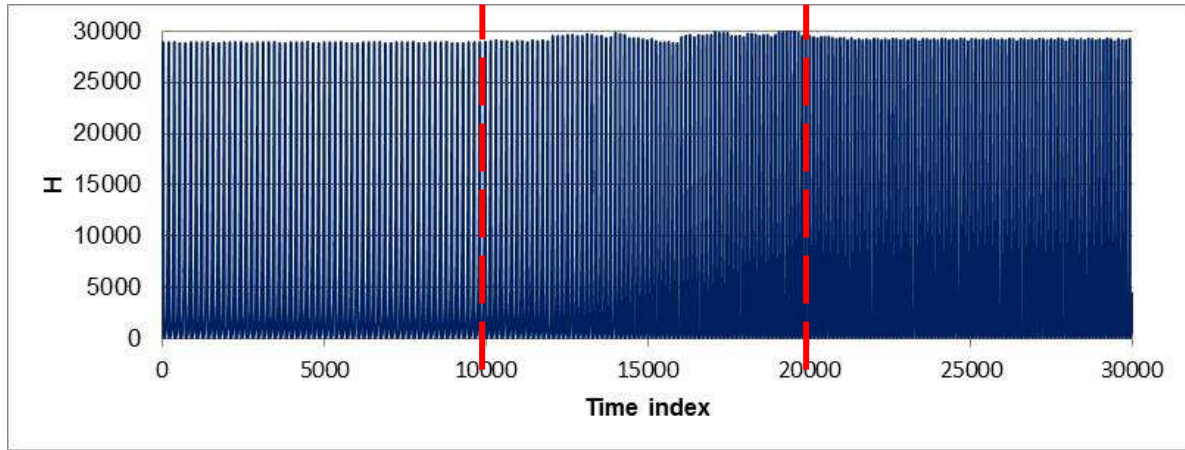


Figure 5.48: Plot of predator–prey time series.

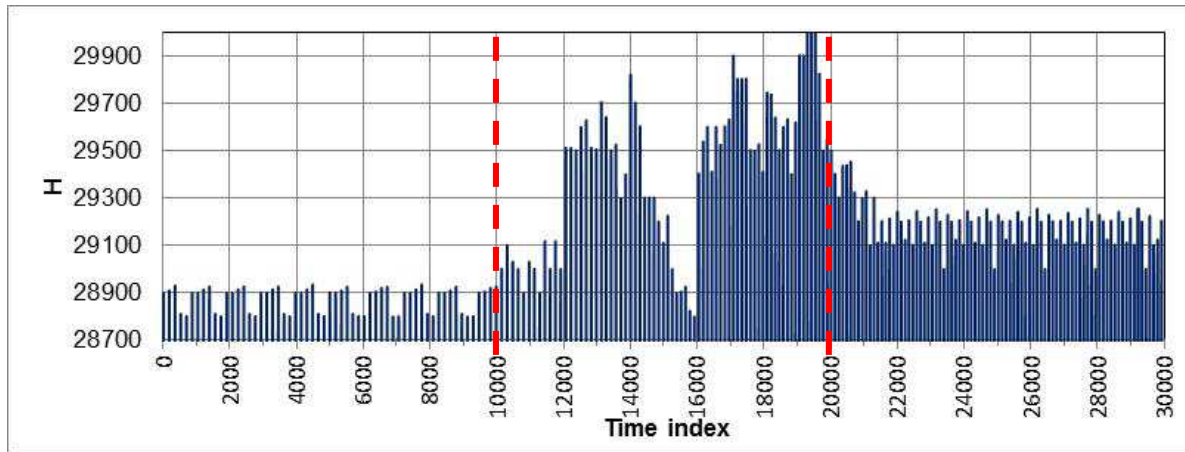


Figure 5.49: Plot of predator–prey time series, showing the top pattern made by the time series.

5.6.1 Calculation of Embedding Parameters

The time series was embedded using an embedding dimension of 9 and a time delay of 22. Average mutual information was used to calculate the time delay, and false nearest neighbour was used to calculate the embedding dimension. Figure 5.50 shows a plot of average mutual information against time delay. The optimum time delay is the first minimum point of the plot of AMI against time lag. Figure 5.51 shows a plot of false nearest neighbour against embedding dimension. The optimal embedding dimension is the first point where the false nearest neighbour reaches zero. See equation B2.1 in the appendices for calculation of the average mutual information and equation C1.7 for false nearest neighbour determination.

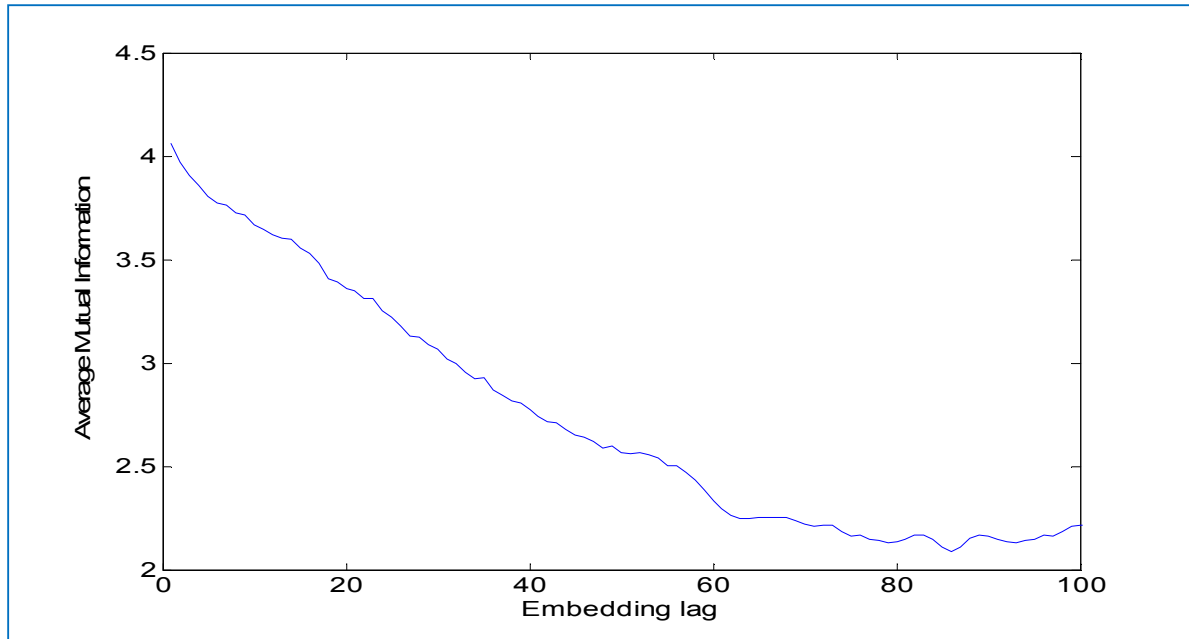


Figure 5.50: Plot of average mutual information against time delay.

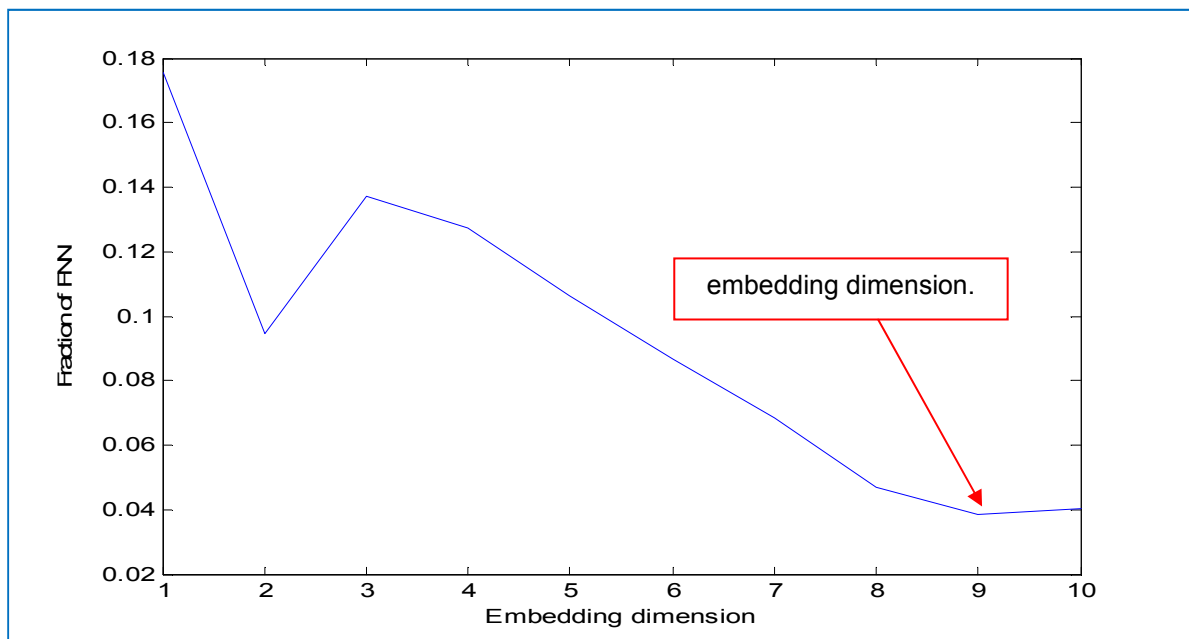


Figure 5.51: Plot of fraction of nearest neighbour against the embedding dimension.

Optimal time delay and embedding dimensions should be able to unlock the attractor; Figure 5.52 shows the attractor drawn from the predator-prey time series.

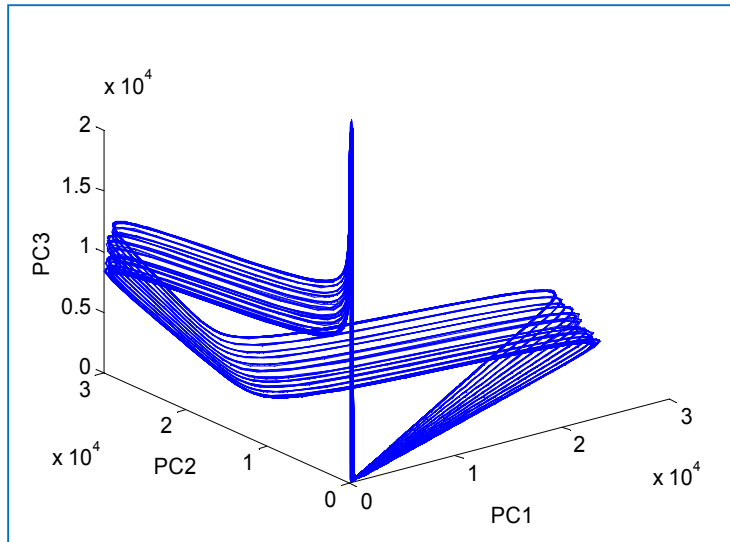


Figure 5.52: Plot of the attractor for the predator-prey time series, with an embedding delay of 22 and an embedding dimension of 9.

5.6.2 Network Training

The first 5 000 data points of the predator-prey process were used for training the network. Ensembles of networks were trained and the best one in each configuration was chosen. Table 5.10 shows networks with different configurations and their performances. The network performance improved with increases in the number of nodes for the mapping / de-mapping layers.

These increases stopped when the number of nodes for the mapping / de-mapping reached seven, an increase above that did not improve the network performance. The training set had more data and probably more complex compared to the other time series used in this work. That is why the network needed more nodes to minimise on underfitting.

Table 5-10: Networks with different configurations and their performances for the predator-prey system.

Network structure			MSE error	Regression			FEV
Mapping layer	Bottleneck layer	De-mapping layer		Training	Validation	Testing	
3	1	3	0.077	0.86	0.83	0.82	0.51
4	1	4	0.065	0.86	0.85	0.84	0.58
5	1	5	0.063	0.86	0.85	0.86	0.58
6	1	6	0.031	0.93	0.93	0.93	0.78
7	1	7	0.013	0.97	0.97	0.97	0.92
7	2	7	0.008	0.98	0.98	0.98	0.95
8	2	8	0.004	0.99	0.99	0.99	0.97
9	2	9	0.003	0.99	0.99	0.99	0.97
10	2	10	0.002	1	1	1	0.97
11	2	11	0.002	1	1	1	0.98
11	3	11	0	1	1	1	1

Figures 5.53 and 5.54 show a plot of the predator-prey time series together with the time series simulated by the trained network. Figure 5.53 is a plot of a segment taken from the first 10 000 observations, which is before the parameter change but not part of the training set. The original time series and the simulated time series lie directly on top of each other showing the excellent accuracy of the network.

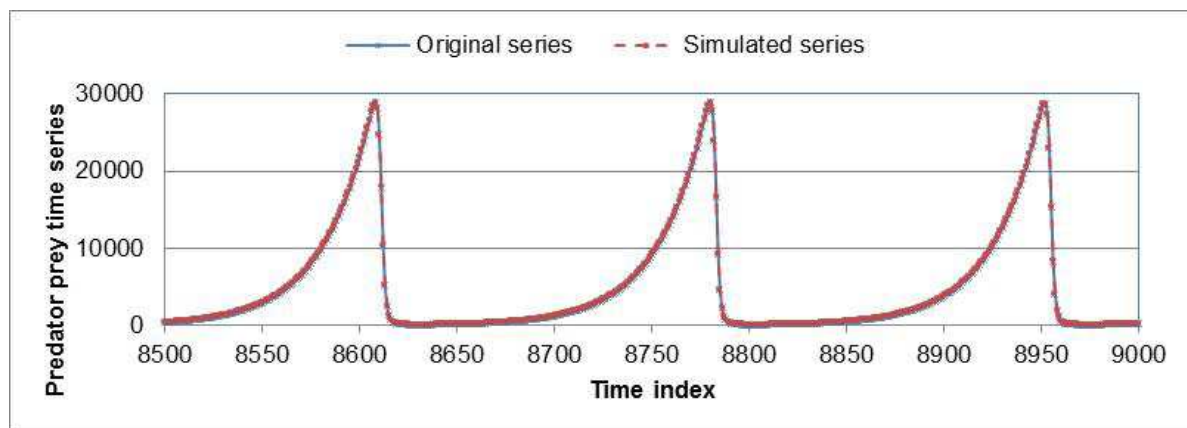


Figure 5.53: A snapshot of the predator-prey time series together with the predator-prey time series simulated using the trained network. The snapshot is taken before the parameter change.

Figure 5.54 is a plot of the predator-prey time series taken from the last 10 000 observations, which is the segment after the parameter change. The original series and the simulated series do not quite match each other. This is because the network was trained with data before the parameter change, and the series generated by the system after the parameter change differs.

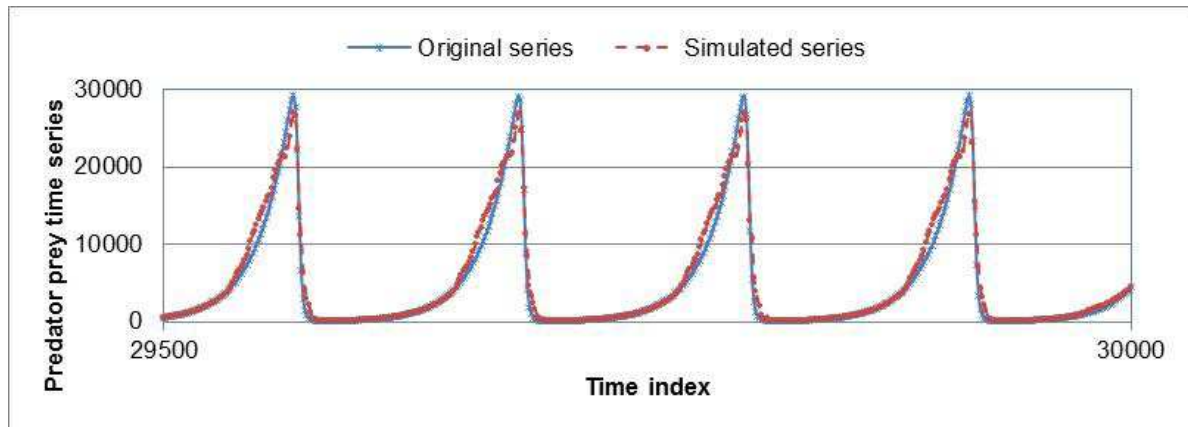


Figure 5.54: A snapshot of the predator-prey time series together with the predator-prey time series simulated using the trained network. The snapshot is taken after the parameter change.

5.6.3 Change-Point Detection for Predator-Prey System with NLSSA

Nonlinear singular spectrum analysis was able to detect change at the 10 000 time index as shown by Figure 5.55. Figure 5.56 shows the first 3000 index points of Figure 5.55 plotted together with the moving average. It shows the multiple false alarms from the change-point detection. One way of minimising the false alarms is to apply a moving average on the residuals before applying the test statistics, also shown in Figure 5.56.

The second method will be to incorporate running length into the change-point algorithm. The running length is the number of consecutive points above the threshold value before confirming a change-point. Figure 5.57 is a plot of the moving average of the squared residuals, it clearly shows the step increments of the parameter changes from time index 10 000 till 20 000.

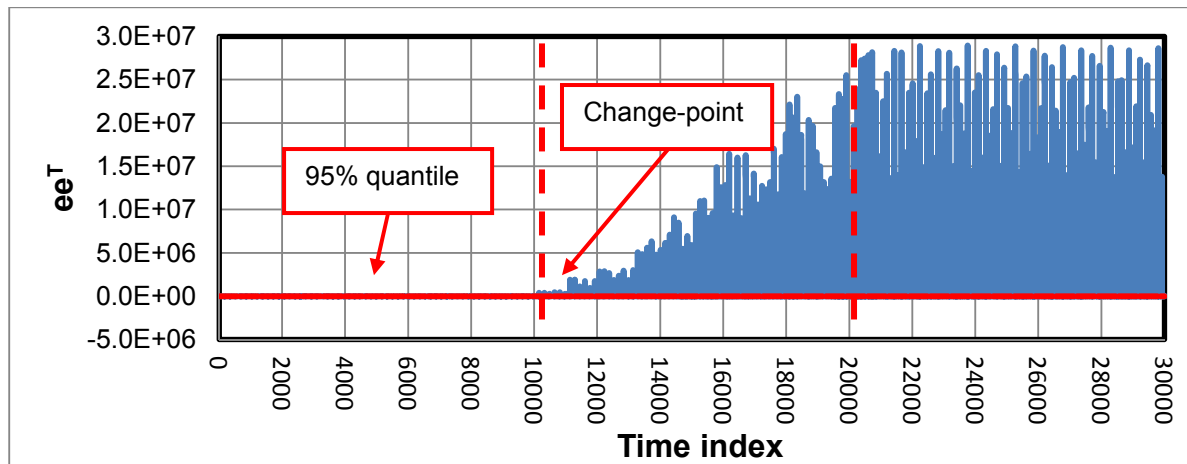


Figure 5.55: Change-point detection for a predator-prey system with NLSSA.

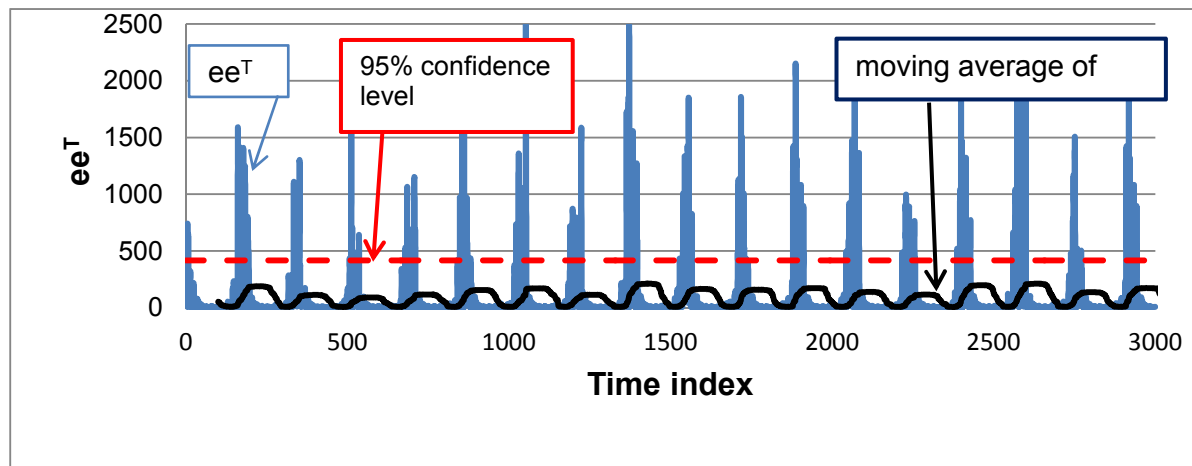


Figure 5.56: Change-point detection for a predator-prey system with NLSSA – the first 3000 observations.

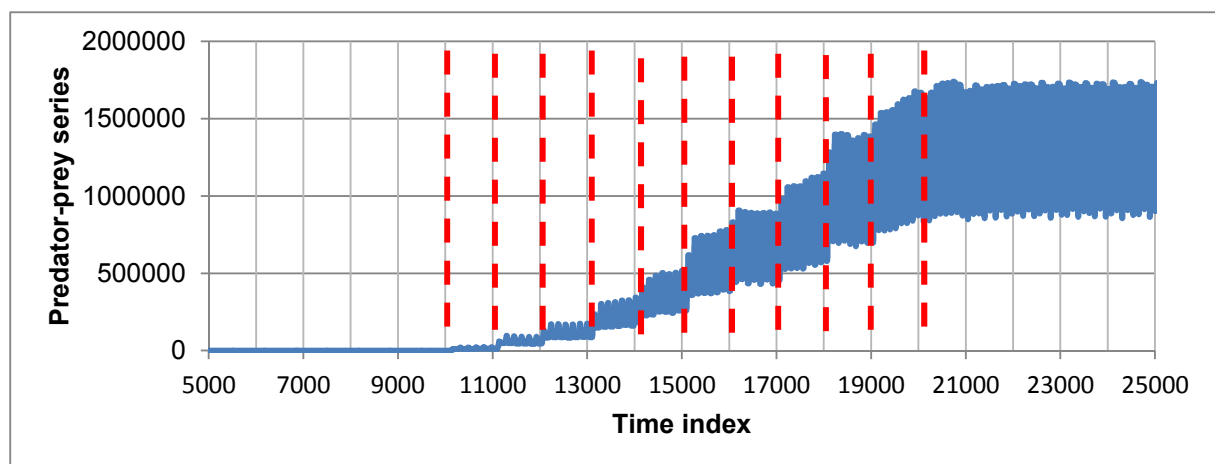


Figure 5.57: Moving average for the squared residuals from the predator-prey system.

5.6.4 Change-Point Detection for a Predator-Prey System with LSSA

A linear singular spectrum analysis change–point detection algorithm was applied to the autocatalytic time series. The following parameters were used in the algorithm: # change-point parameters: $M=12$, Center (no), PC: 1-2, $N=115$, $p=102$, $q=138$. Figure 5.58 shows the result obtained from the LSSA algorithm. LSSA was able to detect change–point at time index 10 300. The technique failed to show the step increments at every 1000th observation from the 10 000th time index until the 20 000th time index where the parameters are kept constant until the 30 000th time index.

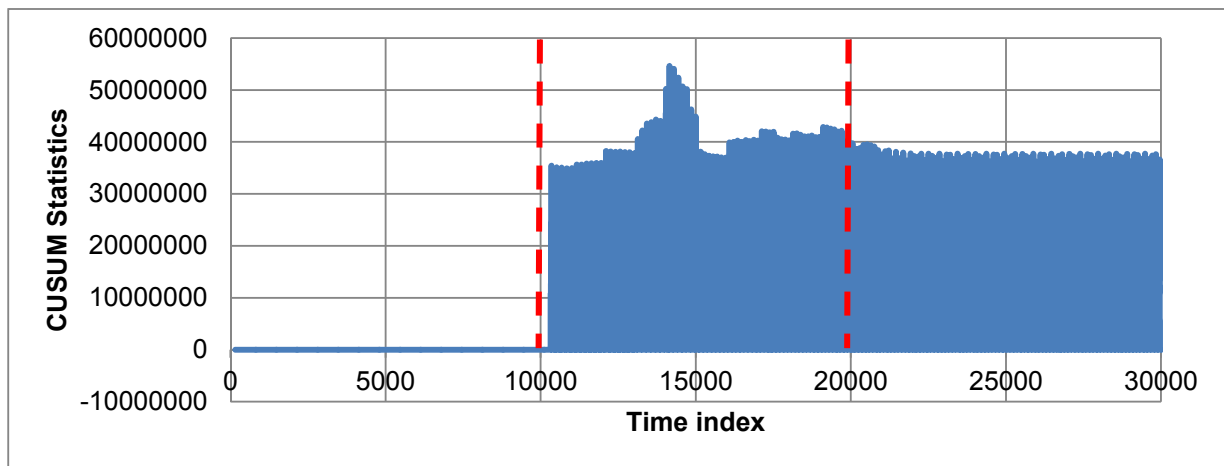


Figure 5.58: Change-point detection for a predator-prey system with LSSA.

5.6.5 Comparison between NLSSA and LSSA

The performance of both NLSSA and LSSA were comparable for the predator-prey system. Table 5.11 shows the fraction of explained variance for the two techniques. Both techniques were able to detect the parameter changes and they both have equal FEV.

Table 5-11: FEV for NLSSA and LSSA

Eigennumber	FEV (NLSSA)	FEV (LSSA)
1	0.92	0.92
2	0.98	0.99
3	1	1

5.7 Comparative Analysis of Change-Point Detection Techniques

Three different techniques have been used for change-point detection on these time series. These techniques are; nonlinear singular spectrum analysis, linear singular spectrum analysis and random forest. Table 5.12 below shows the performances of all three techniques.

Table 5-12: Comparison of the performance of the different change-point detection techniques used for this time series.

Time series	NLSSA		Random Forest		LSSA	
	Detection status	Detection point	Detection status	Detection point	Detection status	Detection point
Gaussian data – mean shift	detected	250	detected	260	detected	250
Gaussian data – variance shift	detected	250	detected	255	detected	250
Autoregressive process	not detected		not detected		not detected	
Two cross-correlated time series	detected	250	not detected		detected	250
BZ- reaction	detected	1200	detected	1300	detected	1300
Autocatalytic reaction system	detected	1200	not detected		not detected	
Predator–prey system	detected	10000	not detected		detected	12000

All the techniques were able to detect change-points for the mean and variance shift in the Gaussian data and they all failed to detect the change-point in the autoregressive process. NLSSA and LSSA techniques were able to detect change-point for the cross correlated time series, whilst Random forest was unable to find the change-point for the same. All the techniques were able to detect change for the BZ reaction system. Although random forest was able to detect the change-point, it was only able to detect it at the 1300th observation when the actual parameter change happened at the 1000th observation. NLSSA was able to detect the change-point for the autocatalytic reaction system, whilst the random forest and LSSA failed to detect the change-point. Overall NLSSA performs better than both random forests and LSSA.

Future work should look at combining two or more change-point detection methods (hybrid change-point detection methods) to get a more robust method. The generation of residuals using auto-associative neural networks yields a series that is uncorrelated; the change-point resembles that of a mean shift. Using a threshold value calculated from the ‘in control’ segment of the time series, it fails to pick up the change when the shift moves from a higher value to a lower value (decrease). Combining two methods will help in this regard.

Time series is first embedded before change-point detection methods are applied to them, investigating the use of neural networks to embed the time series before applying the NLSSA might improve its performance.

5.8 Practical Considerations

Although the proposed nonlinear singular spectrum analysis change-point detection scheme is potentially applicable to different types of data as encountered in practical environments, a critical consideration is effect of data contamination with stochastic elements. In particular, industrial systems are subject to the effect of various systematic and random perturbations that are reflected in the data as noise. In principle, auto-associative neural networks can also be used as low-pass frequency filters, especially where the noise effects can be reliably estimated. Unfortunately, noise estimation is inherently a difficult problem. Specifically, when the noise level is higher than the change that will result from the parameter change any change-point detection is bound to fail because of overshadowing of the residual space used to detect changes. Filtering the high-frequency components may be used as first step prior to change-point detection.

Another limitation is that change-point detection techniques assume that the training data is ‘in-control’ whereas virtually all industrial processes are non-stationary as a result of continuously changing process conditions. Hence, it is necessary to continuously update the ‘in-control’ region as process conditions change.

Chapter 6 Conclusions

In the last few decades, there has been a lot of research interest focussed on change-point detection in view of its importance in industrial and other applications. Most of the focus has been directed on linear approaches. However, practical data are invariably nonlinear due to the dynamic and nonlinear characteristics of underlying phenomena. In this study, a robust nonlinear change point detection scheme based on singular spectrum analysis was proposed, whose performance was evaluated on various simulated time series data with both large and small parameter changes. The method is based on the original singular spectrum analysis method of Moskvina and Zhigljavsky (2003), the key difference being the use of auto-associative neural networks instead of principal component analysis in feature detection. The change-point detection methods were compared on the basis of the fraction of explained variance (FEV), similar in spirit to the eigenvalues of a covariance matrix that explains the percentage of energy (or variance) captured by retained components when using PCA (Kerschen and Golinval, 2003).

Two types of time series data were used for change-point detection: data generated according to a normal or Gaussian probability distribution and data obtained from nonlinear systems. For the Gaussian data, no performance improvement was observed when using the proposed nonlinear SSA approach compared to the original SSA method. Three time series were simulated from the following nonlinear systems: (i) Belousov-Zhabotinsky (BZ) reaction, (ii) autocatalytic reactor, and (iii) the predator- prey series. The proposed nonlinear method was able to detect simulated change-points across all three time series, while the linear SSA method could only detect change-points on the BZ reactions and the predator-prey system. On the basis of the FEV measure, nonlinear SSA performed better than LSSA on the BZ, but linear SSA performed better on the autocatalytic system. Comparable performances were observed for both techniques on the predator-prey system.

In general, nonlinear SSA-based approach does not offer any advantages over linear SSA when the data follow a linear generating mechanism. When the time series is nonlinear, auto-associative neural networks offer advantages over the linear SSA. Hence, any application of the proposed method requires nonlinearity testing prior to analysis. As with most data-driven approaches, there is no one change-point

detection technique that works on all types of data; different techniques performs better on different types of data.

Appendices

A. CUSUM Algorithm

The detection of a change-point is performed by the cumulative-sum (CUSUM) algorithm on the generated squared residuals $r(i)$. The principle of CUSUM stems from a stochastic hypothesis testing method (Chen, 1999).

$$\text{Cumulative sum: } S_i = \sum_{i=1}^k s(r(i)) = \sum_{i=1}^k \ln \frac{p_{\mu_1}(r(i))}{p_{\mu_0}(r(i))} \quad A1$$

In this expression, k denotes the present time instant; $r(i)$ is the residual signal; μ_0 and μ_1 are the mean values under the two hypothesis H_0 and H_1 , assuming that the variance σ^2 is the same. $s(r(i))$ is a log-likelihood ratio of the residual.

A1. Principle of the CUSUM algorithm

The diagnostic signal $r(i)$ has the nature of the Gaussian distribution, due to inclusion of measurement noises that normally obey the Gaussian distribution. Residuals generally are stochastic. Thus, according to this reasonable assumption, the residual $r(i)$ has the Gaussian probability density with mean μ and variance σ :

$$p_{\mu}(r) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(r - \mu)^2}{2\sigma^2}\right) \quad A2$$

The residual signal $r(i)$ carries the information of whether there is a change-point or not. The following hypothesis can be used to evaluate the residuals:

$$H_0: \mu = \mu_0 \text{ and } H_1: \mu = \mu_1 \quad A3$$

A zero hypothesis H_0 is for the case where there is no change, and an alternative hypothesis H_1 for the case where there is a change-point. The log-likelihood ratio of the residual $r(i)$ is defined as:

$$s(r(i)) = \sum_{i=1}^k \ln \frac{p_{\mu_1}(r(i))}{p_{\mu_0}(r(i))} \quad A4$$

Substituting the expressions of $p_{\mu_1}(r(i))$ and $p_{\mu_0}(r(i))$ from equation A2 into equation A4 yields

$$s(r(i)) = \ln \frac{\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(r(i) - \mu_1)^2}{2\sigma^2}\right)}{\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(r(i) - \mu_0)^2}{2\sigma^2}\right)}$$

$$s(r(i)) = \frac{\mu_1 - \mu_0}{\sigma^2} \left(r(i) - \frac{\mu_1 + \mu_0}{2} \right) \quad A5$$

The next step is to calculate the accumulative sum equation A1

$$S_i = \sum_{i=1}^k s(r(i))$$

The valid test consists of the following decisions:

if $g_i \leq h$, accept H_0 ;

if $g_i > h$, accept H_1

where the threshold h is a positive number. g_i is the decision function with the expression as

$$g_i = \begin{cases} S_i, & S_i > 0 \\ 0, & S_i < 0 \end{cases} \quad A6$$

An efficient way to implement the decision function g_i is to use the recursive form that can be written into a maximum-choosing function as follows:

$$g_i = \max(g_{i-1} + s(r(i)), 0) \quad A7$$

The idea of setting a threshold is motivated because g_i will increase when a change-point occurs and decrease when the change disappears. Only the contributions to the cumulative sum that add up to the threshold h must be taken into account in order to determine the decision g_i . During the no change period, the value of g_i is around zero. An alarm is set at the time t_a when the fault is detected.

$$t_a = i, \text{ when } g_i > h \quad A8$$

A2. Enhanced Algorithm for the CUSUM Test

In the real system, the change-point may happen either with a positive μ_1 or a negative μ_1 . The CUSUM algorithm is improved to test the residual signals in two sides – both positive and negative:

$$H_0: \mu = \mu_0 = 0; \mu = \mu_1 = \pm v \quad A9$$

This means that the zero hypothesis is still μ_0 and the alternative hypothesis is two-sided $\mu_1 = \pm v$.

In the two-sided CUSUM algorithm, the decision functions are two equations - one for the positive mean of the residual and the other for the negative mean of the residual signal, both under a change-point.

$$g_i^+ = \max(g_{i-1}^+ + s(r(i)), 0), \text{ where } s(r(i)) = \frac{v}{\sigma^2} \left(r(i) - \frac{v}{2} \right) \quad A10$$

$$g_i^- = \max(g_{i-1}^- + s(r(i)), 0), \text{ where } s(r(i)) = -\frac{v}{\sigma^2} \left(r(i) - \frac{v}{2} \right) \quad A11$$

It is noted that the decision is made when the prior fault happens. This means that both g_i^+ and g_i^- will increase when the change-point occurs; however, the detection time depends on which reaches the threshold earlier.

$$t_a = \min(i : (g_i^+ \geq h) \vee (g_i^- \geq h)) \quad A12$$

where t_a is the time of detecting the fault and an alarm is set. h is the threshold both for g_i^+ and g_i^- .

A3. Initialization of Parameters in the CUSUM Test

Known from the algorithm presented above, μ_0 , μ_1 , σ and h are the parameters designed before the test. First, the variance σ^2 has been assumed the same in both hypotheses. For the no change-point case H_0 , the mean μ_0 and the variance σ^2 can be calculated from the residual signal $r(i)$. Normally, μ_1 can be chosen from the change of the residual r_i with respect to the time when the change happens.

The threshold h can be chosen either by an empirical value if drawing the figure of g_i or by a theoretical method called an average-run-length (ARL) function. In the CUSUM algorithm, the theoretical method aims at make a trade-off between the time interval of two change-point detections and the interval time of two false alarms. One

can expect the shortest interval between two detections of change-points and longest interval between two false alarms. The ARL function is the expected value of the alarm time instant

$$L = E(t_a) \quad A13$$

where L is the value of ARL function and t_a is the alarm time defined in the equation A12 for the enhanced CUSUM algorithm.

An approximation for the ARL function calculates the mean μ_s and the variance σ^2 of the cumulative sum increment $s(r(i))$

$$\text{when } H_0 : \mu = \mu_0, E_{\mu_0}(r(i)) = \mu_0 \text{ then } \mu_s = E_{\mu_0}(s(r(i))) = -\frac{\mu_1 - \mu_0}{2\sigma^2}$$

$$\text{when } H_0 : \mu = \mu_1, E_{\mu_1}(r(i)) = \mu_1 \text{ then } \mu_s = E_{\mu_1}(s(r(i))) = \frac{\mu_1 - \mu_0}{2\sigma^2}$$

In another way, the mean μ_s is written as

$$\mu_s = \begin{cases} \mu_s = E_{\mu_0}(s(r(i))) = -\frac{\mu_1 - \mu_0}{2\sigma^2} & \text{when } H_0: \mu = \mu_0 \\ \mu_s = E_{\mu_1}(s(r(i))) = \frac{\mu_1 - \mu_0}{2\sigma^2} & \text{when } H_1: \mu = \mu_1 \end{cases} \quad A14$$

In both cases, the variance σ_s^2 is derived from

$$\sigma^2 = \frac{(\mu_1 - \mu_0)^2}{2\sigma^2} \quad A15$$

A4. Implementation of the CUSUM Algorithm

The CUSUM algorithm for detection of a change in the mean of a Gaussian distributed residual signal $r(i)$ can be implemented as follows:

At each sample time i

1. Acquire the new data of the residual signal $r(i)$.
2. Compute g_i using equation A7 for a one-sided hypothesis or equations A10 and A11 for a two sided hypothesis.
3. Issue an alarm if $g_i (g_i^+ \text{ or } g_i^-) > h$ at time t_a and initialize the decision function g_i (or g_i^+ and g_i^-) back to zero.

A5. Initialization

1. The mean μ_0 and variance σ^2 of the residual signal are calculated in the nominal working mode.
2. The mean μ_1 of the residual signal can be chosen based on the experiment data of the residual signal under a fault.
3. The threshold h is estimated to meet either the specified mean time for detection τ or the specified mean time between false alarms T .

B. Estimation of Embedding Parameters

Determination of embedding parameters, namely time delay and embedding dimension, is the most important step in nonlinear time series modeling. Various techniques have been developed for estimating embedding parameters, m and τ .

B1. Linear Autocorrelation Function

From basic mathematics, the autocorrelation function, ACF, compares linear dependence of two time series separated by delay and is defined (Cellucci, C J et al. 2003)

$$ACF = C_T = \frac{\sum_{i=1}^{N-\tau} [x_{i+\tau} - \bar{x}][x_i - \bar{x}]}{\sum_{i=1}^{N-\tau} [x_i - \bar{x}]^2} \quad (B1.1)$$

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i \quad (B1.2)$$

The delay time τ is taken where linear independence is found and that is where C_T equals one-half or zero or the first inflation point.

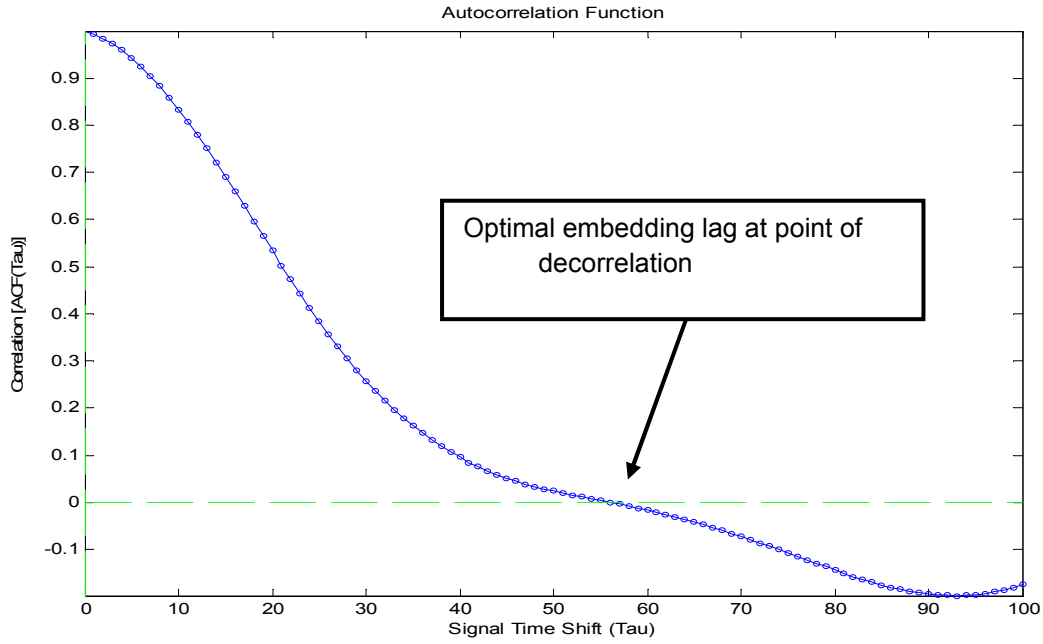


Figure B1: Graph showing the plot of correlation vs time shift allowing the determination of time lag.

B2. Average Mutual Information (AMI)

AMI measures the general dependency of two variables by quantifying the amount of information known about the state $y(n + \tau)$ given that the state $y(n)$ is known (Fraser and Swinney, 1986).

$$I(\tau) = \sum_{y(n), y(n+\tau)} P(y(n), y(n + \tau)) * \log_2 \frac{P(y(n), y(n + \tau))}{P(y(n))P(y(n + \tau))} \quad (\text{B2.1})$$

where $P(y(n), y(n + \tau))$ is the joint probability distribution for the two measurements $y(n)$ and $y(n + \tau)$. If τ was too big $y(n)$ and $y(n + \tau)$ would be completely independent in a nonlinear fashion, then $P(y(n), y(n + \tau)) = P(y(n))P(y(n + \tau))$ and the mutual information is zero. For τ too small, $P(y(n)) = P(y(n + \tau))$. Optimal τ should be in such a way that the measurements are somewhat but not totally independent. Fraser suggested the use of the function $I(\tau)$ as a kind of autocorrelation function to determine optimal delay time τ . The minimum point of the function $I(\tau)$ shows a point where sample points are maximally decor related. The optimal delay time τ is the first minimum of function $I(\tau)$ (Fraser and Swinney, 1986).

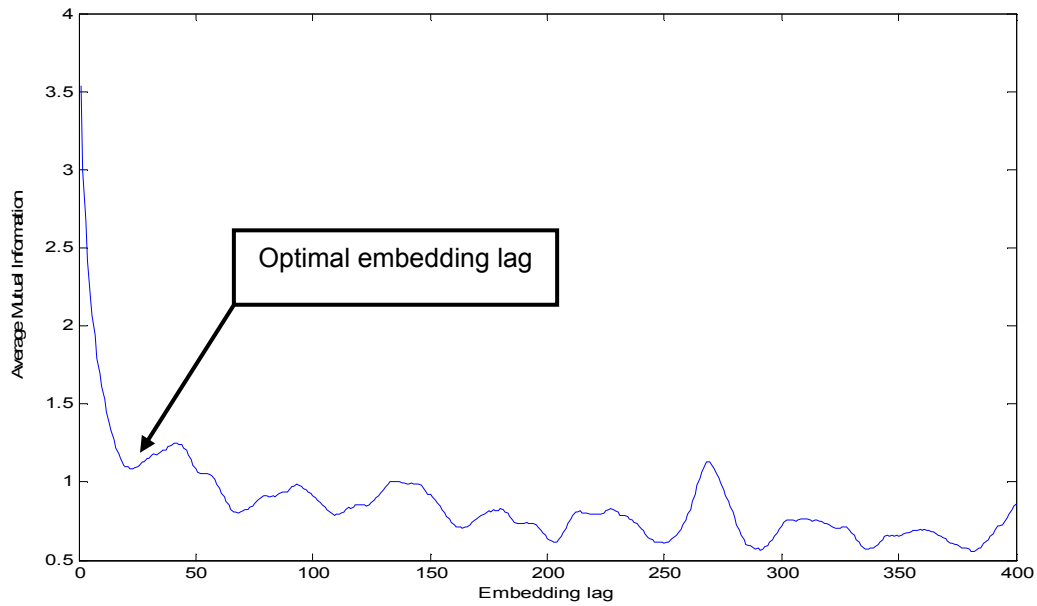


Figure B2: Graph showing the plot of average mutual information vs embedding lag.

B3. Embedding Dimension

Every m dimensional manifold can be embedded within an embedding space of dimension $d = 2m + 1$ (Takens, 1981). Therefore, an attractor may, in general, be reconstructed within the embedding space of dimension $d = 2m + 1$, where m is the dimension of the state space of the underlying real attractor.

Takens delay embedding theorem was generalized by Sauer et al (1991) who showed that the dimension need only $d = 2m$, if there are no orbits with period Δt or $2 \Delta t$ and only a finite number of orbits with higher periods. Δt is the time lag between data samples.

The Lorenz attractor can be embedded by the time-delay method in $d = 3$. This is in contrast to the theorem which suggests that $d = 5$ (Kennel et al., 1992). Working in dimensions larger than the minimum required by the data leads to excessive computation when investigating other subsequent questions, e.g. Lyapunov exponents or predictions (Kennel et al., 1992). It is therefore important that in the determination of embedding dimensions, methods that will give the minimum embedding dimension required to unlock the attractor be used. Several methods

have been addressed in literature for determining the embedding dimension from an observed time series.

False Nearest Neighbours (FNN)

False Nearest Neighbors states that if the dynamics of a system is to be reconstructed successfully in R^d , then all the neighboring points in R^d would also be neighbors in R^{d+1} (Kennel and Abarbanel, 2002). This method checks neighbors in successively higher embedding dimensions until only a negligible number of false neighbors are found when increasing the dimension from R^d to R^{d+1} . This dimension is chosen as the embedding dimension.

The algorithm is as follows (Manabe and Chakraborty, 2006). Let the delay coordinate vector in the m dimension be

$$u(t) = (y(t), y(t + \tau), \dots, y(t + (m - 1)\tau)) \quad (C1.1)$$

and denote the nearest vector for $u(t)$ by

$$u^{NN}(t) = (y^{NN}(t), y^{NN}(t + \tau), \dots, y^{NN}(t + (m - 1)\tau)) \quad (C1.2)$$

Then, the Euclidean distance between $u(t)$ and $u^{NN}(t)$ in the m dimension is

$$R_m(t) = \sqrt{\sum_{k=1}^m \{y(t + (k - 1)\tau) - y^{NN}(t + (k - 1)\tau)\}^2} \quad (C1.3)$$

On the other hand, the Euclidean distance between $u(t)$ and $u^{NN}(t)$ in the $m+1$ dimension is

$$R_{m+1}(t) = \sqrt{\sum_{k=1}^{m+1} \{y(t + (k - 1)\tau) - y^{NN}(t + (k - 1)\tau)\}^2} \quad (C1.4)$$

The relative distance between $u(t)$ and $u^{NN}(t)$ in the $m+1$ dimension is

$$R_L = \sqrt{\frac{R_{m+1}(t)^2 - R_m(t)^2}{R_m(t)^2}} \quad (C1.5)$$

$$R_L = \frac{|y(t + m\tau) - y^{NN}(t + m\tau)|}{R_m(t)} \quad (C1.6)$$

When R_L is larger than a heuristic threshold, we have a false neighbor. The optimal embedding dimension, which is sufficiently high for reconstructing the attractor, is the dimension in which R_L is zero or sufficiently small.

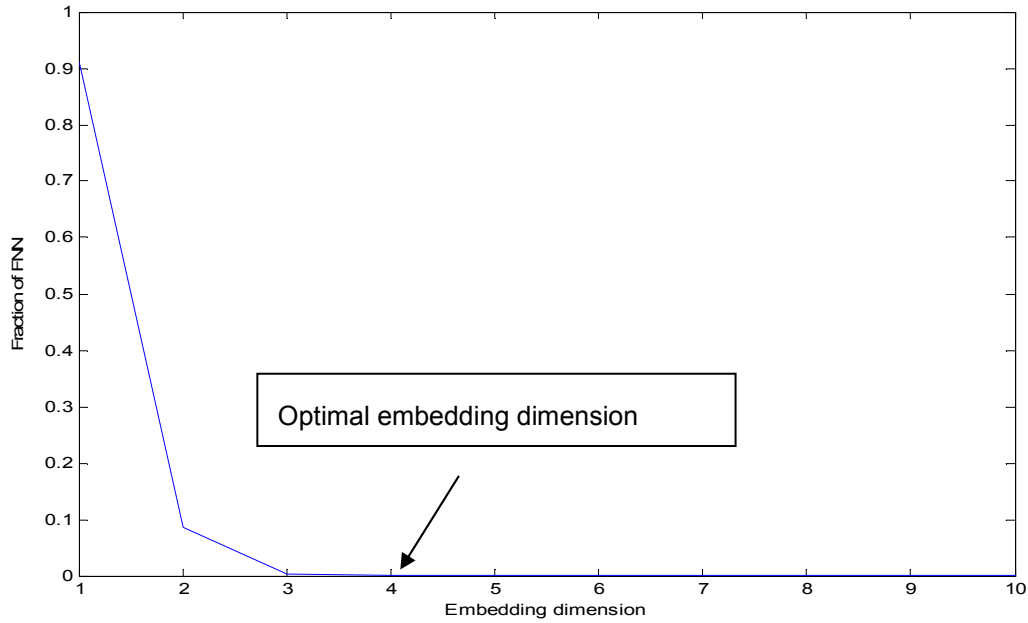


Figure C1: Plot of FNN against the embedding dimension

Correlation Dimension

To determine the minimum dimension of the embedding space a correlation dimension is calculated for the reconstructed attractor. The correlation dimension is calculated from the power law relation between the correlation integral, C , of an attractor and neighborhood radius, r . Correlation dimension is the slope of the log plot of C versus r (Zarghami et al, 2008).

The following Algorithm is used to calculate correlation dimension (D_2) (Grassberger and Procaccia, 1983a)

$$D_2 = \lim_{r \rightarrow 0} \frac{\log C(N, r)}{\log(r)} \quad (C2.1)$$

Where correlation integral $C(N, r)$ is defined by:

$$C(N, r) = \frac{2}{N(N-1)} \sum_{i,j(1 \leq i < j \leq N)} H(r - \|x_i - x_j\|) \quad (C2.2)$$

where $H(x)$ is the Heaviside step function, defined as 1 if $y > 0$ and 0 if $y < 0$, where $y = (r - \|x_i - x_j\|)$. The summation counts the number of pairs (x_i, x_j) for which the distance $\|x_i - x_j\|$ is less than r . r is a radius around each reference point x_i for small values of r , and N is the number of sample points on the attractor. For big N and small, $C(N, r) \approx r^{D_2}$. A plot of $C(N, r)$ versus r on a log – log scale will give a slope that is equal to the correlation dimension, D_2 .

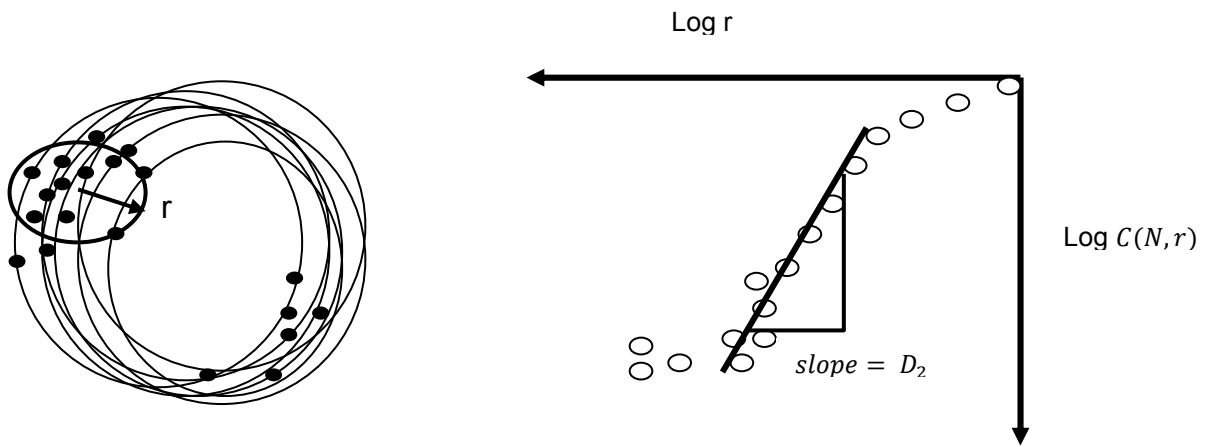


Figure C2: Correlation dimension calculation.

References

1. Abarbanel HD. *Analysis of observed chaotic data* (1st edition). New York: Springer-Verlag, 1996.
2. Abdella M, Marwala T. Treatment of missing data using neural networks and genetic algorithms. In: *Proceedings of International Joint Conference on Neural Networks*. Montreal, 2005: 598-603.
3. Abdi H, Williams LJ. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2, 2010: 434-459.
4. Addison PS. *Fractals and chaos: An illustrated course*. Institute of Physics, Bristol, Philadelphia, 2001.
5. Aldrich C. Cluster analysis of mineral process data with auto associative neural networks. *Chemical Engineering Communications*, 2000: 121-137.
6. Aldrich C. *Exploratory analysis of metallurgical process data with neural network and related methods*. Netherlands: Elsevier Science, 2002.
7. Aldrich C, Jemwa GT. Detecting change in complex process systems with phase space methods. *The 2007 Spring National Meeting. 2007 – AIChE*, 2007: 1-10.
8. Andrienko, N, Gennady A. *Exploratory Analysis of Spatial and Temporal Data: A Systematic Approach*. Berlin: Springer-Verlag, 2006.
9. Barnard JP, Aldrich C, Gerber M. Embedding of multidimensional time-dependent observations. *Physical Review E* 64, 2001: 1- 4.
10. Basseville M, Nikiforov I. *Detection of abrupt changes: theory and application*. Englewood Cliffs, N.J.: Prentice Hall, 1993.
11. Behbahani RM, Jazayeri-Rad H, Hajmirzaee S. Fault detection and diagnosis in a sour gas absorption column using neural networks. *Chem. Technol* 32, no. 5. 2009: 840-845.
12. Beneki C, Bruno E, Costas L. Signal extraction and forecasting of the UK tourism income time series: A singular spectrum analysis approach. *Journal of Forecasting*. DOI:10.1002/for.1220, 2011.
13. Bishop CM. *Neural Networks for Pattern Recognition*. New York: Oxford University Press, 1995.
14. Blazek RB, HongJoong K, Boris R, and Alexander T. A novel approach to detection of “denial-of-service” attacks via adaptive sequential and batch-sequential change-point detection methods. In: *Proceedings of the 2001 IEEE*, 2001: 220-226.
15. Botha JP. *Detecting change in complex process systems with phase space methods*. Masters Thesis, University of Stellenbosch, South Africa, 2006.
16. Bratina BN, Muskinja, Ovornik BT. Design of an auto-associative neural network by using design of experiments approach. *Neural Computing and Applications*, 19, 2010: 207-218.
17. Breiman L. Bagging predictions. *Machine Learning*, 24(2), 1996:123-140.
18. Bulsari, AB. *Neural network for chemical engineers*. Amsterdam: Elsevier, 1995.
19. Cannon M, Deshmukh V, Kouvaritakis B. Nonlinear model predictive control with polytopic invariant sets. *Automatica* 39. 2003: 1487-1494.

20. Cellucci CJ, Albano AM, Rapp PE. Comparative study of embedding methods. *Physical Review*, 2003: 1-12.
21. Chen Y, Hwang K, and Wei-Shinn K. Distributed change-point detection of DDoS attacks: Experimental results on DETER test bed. *DETER Community Workshop on Cyber Security Experimentation and Test*. Boston: USENIX Association Berkeley, 2007.
22. Cheon S, and Kim J. Multiple change-point detection of multivariate mean vectors with Bayesian approach. *Computational Statistics and Data Analysis* (Elsevier BV) 54 ,2009: 406-415.
23. Clements MP, Franses PH, Swanson NR. Forecasting economic and financial time-series with nonlinear models. *Forecasting Economic and Financial Time Series Using Nonlinear Methods*, 20(2), 2004: 169-183.
24. Coulibaly P, Anctil B, Bobee B. Daily reservoir inflow forecasting using artificial neural networks with stopped training approach. *Journal of Hydrology* 230, 2000: 244-257.
25. Dana ED. Rapid assessment of population trends of invasive species: Singular Spectrum Analysis(SSA). *Management of Biological Invasions*, 2010: 34-36.
26. Daszykowski M, Walczak B, Massart DL. A journey into low-dimensional spaces with auto-associative neural networks. *Talanta*, 59, 2003: 1095-1105.
27. De Oca VM, Jeske DR, Zhang Q, Rendon C, Marvasti M. A cusum change-point detection algorithm for non-stationary sequences. *The Journal of Systems and Software*, 2010: 1288-1297.
28. Dirion JL, Cabassud M, Casamatta G, Lann MV. Neural network for process control: Application to the temperature control of batch chemical reactors. Edited by T, Leonendes Cornelius. *Expert Systems*, Academic Press, 2002: 443-448.
29. Downey AB. A novel changepoint detection algorithm. *Applied Microbiology and Biotechnology*, 2008: 1-11.
30. Eckmiller R, Malsburg CV. Neural computers. In: Eckmiller R, Malsburg CV, *Neural Computers*. New York: Springer-Verlag, 1988: 291 – 300
31. Findeisen R, Allgöwer R F. An introduction to nonlinear model predictive control. In: *21st Benelux Meeting on Systems and Control*, 2002: 1-23.
32. Fraser AM, Swinney HL. Independent coordinates for strange attractors from mutual information. *Phys.Rev A*, 33, no. 2 ,1986: 1134-1140.
33. Frisén M. Statistical surveillance, optimality and methods. *International Statistical Review*, 71, no. 2 ,2003: 403-434.
34. Galvan IM, Zaldivar JM, Hernandez H, Molga E. The use of neural networks for fitting complex kinetic data. *Computers Chem.Eng*, 20(12), 1996: 1451-1465.
35. Gavit, Patrick, Baddour Y, Tholmer R . Use of change-point analysis for process monitoring and control. *BioPharm International* 22(8),2009 : 46-54.
36. Ghil M, Allen M, Dettinger M, Ide K, Kondrashov D, Mann M, Robertson A, Saunders A, Tian, Y, Varadi F, Yiou P. Advanced spectral methods for climatic times series. *Reviews of Geophysics*, 40(1), 2002:3.1-3.41.
37. Grassberger P. , Procaccia I. Characterization of strange attractors. *Physical Rev. Letters*, 50(5), 1983: 346-349.
38. Habibi RS, Sadooghi-Alvandi M, and Nematollahi AR. Change-point detection in general class of distributions. *Communications in statistics* 34, no. 9 ,2005: 1935-1938.

39. Hajizabath E, Ardakani HD, Shahrabi J . Application of data mining techniques in stock markets:A survey. *Journal of Economics and International Finance*, 2010: 109-118.
40. Hassani H, Zhigljavsky A. Singular spectrum analysis: methodology and application to economics data. *Journal of Systimatic Science and Complexity* 22 , 2006: 372-394.
41. Hassani H. Singular spectrum analysis: methodology and comparison. *Journal of Data Science* 5 ,2007: 239-257.
42. Hill T, Lewicki P. *Statistics: Methods and applications*. Tulser OK, 2007.
43. Himmelblau DM. Accounts of experiences in the application of artificial neural networks in chemical engineering. *Ind. Eng. Chem. Res.*, 47 ,2008: 5782-5796.
44. Himmelblau DM, Karjala TW. Rectification of data in a dynamic process using artificial neural networks. *Computers Chem Engng*, 20(617),1995: 805-812.
45. Hornero R, Alonso A, Jimeno N, Jimeno A, Lopez M. Estimation of correlation dimension to evaluate cognitive performance in schizophrenic patients using a new computer technique. *Non-linear Dynamics, Psychology and Life Sciences*,3(1),1999: 49-63.
46. Hsieh W. Nonlinear multivariate and time series analysis by neural network methods. *Review of Geophysics*, 42, 2004: 1-25.
47. Hsieh w, Benyang T. Applying neural network models to prediction and data analysis in meteorology and oceanography. *Bulletin of the American Meteorological Society*, 79, no. 9 ,1998: 1855-1870.
48. Hussain AM. Review of the applications of neural networks in chemical process control and simulation and online implementation. *Intelligence in Engineering* 13 ,1999: 55-68.
49. Hwarnk, Brian H. Detecting process mean shift in the presence of autocorrelation:a neural-network based monitoring scheme. *International Journal of Production Research*, 2003: 573–595.
50. Jemwa GT, Aldrich C. Kernel-based fault diagnosis on mineral processing plants. *Minerals Engineering* 19 ,2006: 1149-1162.
51. Jemwa GT. *Multivariate nonlinear time series analysis of dynamic process systems*. Masters thesis, University of Stellenbosch, South Africa 2003.
52. Kawahara Y, Sugiyama M. Change-point detection in time-series data by direct density-ratio estimation. *SIAM international Conference*, 2009: 389-400.
53. Kawahara Y, Sugiyama M. Change-point detection in time-series data by direct density-ratio estimation. *In: Proceedings of 2009 SIAM International Conference on Data Mining (SDM2009)*, 2009: 389-400.
54. Kawahara, Yoshinobu, Yairi T, Machida K. Change-point detection in time-series data based on subspace identification. *Seventh IEEE International Conference on Data Mining*. Computer Society IEEE, 2007: 559-565.
55. Keller K, Sinn M. A standardized approach to the Kolmogorov-Sinai entropy. *Nonlinearity* 22 ,2009: 2417-2422.
56. Kennel MB, Abarbanel HD. False neighbors and false strands: A reliable minimum embedding dimension algorithm. *Physical Review E*, 66 ,2002: 1-18.
57. Kennel MB, Brown R, Abarbanel HD. Determining embedding dimension for phase-space reconstruction using geometric construction. *Physical Review A*, 45 , 1992: 3403-3411.

58. Kersschen G, Golival J. Feature extraction using auto-associative neural networks. *Smart Materials and Structures*, 13 , 2004: 211-219.
59. Keviczky T, Balas GJ. Receding horizon control of an F-16 aircraft: a comparative study. *Elsevier Science*, 2005: 1-21.
60. Kia SH, Henao H, Capolino GA, Picardie JV. A high-resolution frequency estimation method for three-phase induction machine fault detection. *Industrial Electronics, IEEE*, 2007: 2305-2314.
61. Kim H, Eykholt SR, Salas JD. Nonlinear dynamics, delay times, and embedding windows. *Physica D*, 127, 1999: 48–60.
62. Kiremire BE, Marwala T. Nonstationarity detection: The use of the cross correlation integral in ECG and EEG profile analysis. *IEEE Computer Society*, 2008: 373-378.
63. Kocijan J, Murray-Smith R, Rasmussen C E, Likar B. Predictive control with Gaussian process models. In: *Proceedings of IEEE Region and Eurocon 2003: Computer as a Tool*, 2003: 352-356.
64. Kramer AM. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2), 1991: 233-243.
65. Lee J, Yoo C, Choi SW, Vanrolleghem PA, Lee I. Nonlinear process monitoring using kernel principal component analysis. *Chemical Engineering Science*, 2004: 223-234.
66. Lerner BH, Guterman, Aladjem M, Dinstein I. Feature extraction by neural network nonlinear mapping for pattern classification. *ICPR13, Vienna 4* , 1996: 320-324.
67. Li C, Zhao W. Stochastic performance analysis of non-feedforward networks. *Telecommunication Systems*, 43, 2010: 237-252.
68. Li, Hongbin, Dakui F, Xiao B. Prediction and elucidation of the population dynamics of *Microcystis* spp. in Lake Dianchi (China) by means of artificial neural networks. *Ecological Informatics*, 2(1), 2007: 184-192.
69. Li YG, Nilkitsaranont P. Gas turbine performance prognostic for condition-based maintenance. *Applied Energy* , 86 , 2009: 2152-2161.
70. Lidia A, Aldrich C. Change point detection in time-series data with random forests. *Control Engineering Practice* , 18, 2010: 990-1002.
71. Lin, Sheng-Ya, Liu J, Zhao W. Adaptive CUSUM for anomaly detection and its application to detect shared congestion. Technical Report 2007-1-2, Texas A&M University.
72. Liu X, Chen L. Complex dynamics of Holling type II Lotka-Volterra predator-prey system with impulsive perturbations on the predator. *Chaos, Solitons and Fractals*, 2003: 311-320.
73. Lund, Robert, Xiaolan LW, Lu Q, Reeves J, Gallagher C, Feng Y. Change-point detection in periodic and autocorrelated time series. *Journal of Climate* (20), 2007: 5178-5190
74. Manabe Y, Chakraborty B. A novel approach for estimation of optimal embedding parameters of nonlinear time series by structural learning of neural network. *Neurocomputing* 70 , 2006: 1360-1371.
75. Martínez OC, Ramaswamy HS, Ayala-Aponte AA. Artificial neural network modeling of osmotic dehydration mass transfer kinetics of fruits, *Drying Technology*, 25(1) , 2007: 85-95.

76. Matthias, Fischer M. Analysis of nonlinear predictive control with extended dynamic matrix control. In: *Proceedings of the 2006. IEEE International Conference on Control Applications Munich, Germany*, 2006: 4-6.
77. Mboup, Mamadou, Join C, and Fliess MI. An online change-point detection method. In: *Proceedings of the Mediterranean Conference on Control and Automation Congress Centre, Ajaccio, France: IEEE*, 2008. 1290-1295.
78. Mei Y. Sequential Change-point detection when unknown parameters are present in the pre-change distribution. *The Annals of Statistics*, 2006: 92-122.
79. Molga EJ, Woezik BA, Westerterp KR. Neural networks for modelling of chemical reaction systems with complex kinetics: oxidation of 2-octanol with nitric acid. *Chemical Engineering and Processing* 39 ,1999: 323-334.
80. Montes D, Veronica, Daniel R, Jeske, Zhang Q, Rendon C, Marvasti M. A cusum change-point detection algorithm for non-stationary sequences with application to data network surveillance. *The Journal of Systems and Software*, 2010: 1288-1297.
81. Morgenthaler S. Exploratory data analysis. *Computer Statistics*, 2009: 33-44.
82. Moskvina V, Zhigljavsky A. An algorithm based on singular spectrum analysis for change-point detection. *Communication in Statistics, Simulation and Computation*, 32, no. 2 ,2003: 319-352.
83. Nakamura T, Luo X, Small M. Testing for nonlinearity in time series without the Fourier transform. *Physical Review E* 72 ,2005: 1-4.
84. Nazario D, Ramirez B, Tep S. Transient detection with an application to a chemical process. *Computers ind. Eng*, 1997: 896-908.
85. Ott E. Chaos in dynamical systems. In: *Chaotic behaviour in systems (2nd Edition)*, Cambridge: University Press, 2002: 1-16.
86. Pesonen E, Eskelinen M, Juhola M. Treatment of missing data values in a neural network based decision support system for acute abdominal pain. *Artificial Intelligence in Medicine* 13 ,1997: 139-146.
87. Ramirez-Beltran ND, Jackson H. Application of neural networks to chemical Processes control. *Computers and Industrial Engineering*, 37, 1999: 387-390.
88. Reeves, Jaxk, Chen J, Wang XL, Lund R, and Lu Q. A review and comparison of changepoint detection techniques for climate data. *Journal of Applied Meteorology and Climatology*, 46 ,2006: 900-915.
89. Rhodes C, Morari M. False-nearest-neighbors algorithm and noise-corrupted time series. *Physica Review E* 55, no. 5 ,1997: 6162-6170.
90. Rosenblatt F. The perceptron: A probabilistic model for in the brain. *Psychological Review*, 1958: 386-408.
91. Rumelhart DE, Hinton GE, Williams R J. Learning representations by backpropagating errors. *Nature* ,323 ,1986: 533-536.
92. Russell EL, Chiang LH, Braatz RD. *Data-driven techniques for fault detection and diagnosis in chemical proceses*. London: Springer-Verlag , 2000.
93. Salgado DR, Alonso FJ. Tool wear detection in turning operations using singular spectrum analysis. *Journal of Materials Processing Technology*, 2006: 451-458.
94. Sandri M. Numerical calculation of Lyapunov exponents. *The Mathematica Journal*, 1996.

95. Sejnowski TJ. Neural computers. In: Eckmiller R , Malsburg CV, *Neural network learning algorithms*, New York: Springer-Verlag, 1989: 291-300.
96. Shi, Xiaoping, Yuehua W, Jin B. A novel approach for fast detection of multiple change points in linear models. *arXiv*, 2011: 1-37.
97. Small M, Tse CK. Optimal embedding parameters: a modelling paradigm. *Physica D* 194 ,2004: 283-296.
98. Smith, Wayne J, Barth RB. Text analytics for homeland security technology development. *IEEE*, 2009: 621-627.
99. Sprott JC. *Chaos and time-series analysis*. New York: Oxford University Press, 2004.
100. Stark JD, Broomhead S, Davies ME, Huke J. Takens embedding theorems for forced and stochastic systems. *Journal of Nonlinear Science* 13 ,1997: 519-577.
101. Staudacher M, Telser S, Amann A., Hinterhuber H, Ritsch-Marte M. A new method for change-point detection developed for on-line analysis of the heart beat variability during sleep. *Physica A*, 2005: 582-596.
102. Strozzi F. Application of nonlinear time series analysis techniques to high frequency currency exchange data. *Liuc Papers no. 99, Serie Metodi Quantitavi* 14, 2002: 1-46.
103. Sun J, Zha Y, Nakamura T, Small M. From phase space to frequency domain: A time-frequency analysis for chaotic time series. *Physical Review E* 76 ,2007: 1-8.
104. Vaisman, Lev, Zariffa J, Popovic MR. Application of singular spectrum-based change-point analysis to EMG-onset detection. *Journal of Electromyography and Kinesiology*, 2010: 750-760.
105. Van Lith P. *Hybrid Fuzzy-First Principles Modeling*. Enschede: University of Twente Press, 2002.
106. Venkatasubramanian V, Rengaswamy R, Yin K, Kavuri S. A review of process fault detection and diagnosis Part I: Quantitative model-based methods. *Computers and Chemical Engineering*, 2003: 293-311.
107. Venkatasubramanian V, Rengaswamy R, Yin K, Kavuri S. A review of process fault detection and diagnosis Part III: Process history based methods. *Computers and Chemical Engineering*, 2003: 327-346.
108. Venkatasubramanian V. Abnormal events management in complex process plants: challenges and opportunities in intelligent supervisory control. *Proceedings of the FOCAPO*. In I.E. Grossmann & C.M. McDonald: CACHE, 2003. 117-132.
109. Verdier G, Hilgert N, and Vila J. Optimality of CUSUM rule approximations in. *IEEE*, 2008: 5102-5112.
110. Vincent L A. A technique for the identification of inhomogeneities in Canadian temperature series. *Journal of Climate Change*, 1998: 1094-1104.
111. Virk S, Muhammad A, Martinez-Enriquez A. Fault prediction using artificial neural network and fuzzy logic. *Seventh Mexican International Conference on Artificial Intelligence*. Mexico: IEEE Computer Society, 2008. 149-154.
112. Vyas R, Sharma K, Prakash O, Scheider Simon. Associative classifiers for predictive analytics: comparative performance study. *Computer Modeling and Simulation*, 2008: 289-294.

113. Wang P, Cox C. Study on the application of auto-associative neural network. *IEEE. ICSP'04 Proceedings Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai*, 2004: 3291-3295.
114. Wang P, Cox C. Study on the application of auto-associative neural network. *CSP Proceedings*, 2004: 1570-1573.
115. Wei X, Hanping H, Yue Y, and Wang Q. Anomaly detection of network traffic based on the largest Lyapunov exponent. *IEEE*, 2010: 581-585.
116. Whitney H. Differentiable manifolds. *Annals of mathematics* 37, no. 3, 1936: 645-680.
117. Wise BM, Gallagher NB. The process chemometrics approach to process monitoring and fault detection. *Elsevier Science*, 1996: 329-348.
118. Wolf A, Swift JB, Swinney HL, and Vastano JA. Determining Lyapunov exponents from a time series. *Physica 16D*, 1984: 285-317.
119. Wythoff BJ. Backpropagation neural networks. *Chemometrics and Intelligent Laboratory Systems* 18 ,1993: 115 – 155.
120. Xie J. *Neuro-Fuzzy Modelling and Model-Based Fault Detection on an Autonomous Vehicle*. Masters Thesis Institute of Electronic Systems Aalborg University, 2005.
121. Yonchev A, Findeisen R, Ebenbauer C, Allgöwer F. Model predictive control of linear continuous time singular systems subject to input constraints . *IEEE* , 2004: 247- 252.
122. Yu DL, and Gomm JB. Implementation of neural network predictive control to a multivariable chemical reactor. *Control engineering practice* 11 ,2003: 1315- 1323.
123. Zakaria Z, A Isa N, Suandi SA. A study on neural network training algorithm fot multiface detection in static images. *World academy of science, Engineering and technology* 62 2010: 170-173.
124. Zarghami R, Mostoufi N, Sotudeh-Gharebagh R. Nonlinear characterization of pressure fluctuations in fluidized beds. *Ind.Eng.Chem.Res* 47 ,2008: 9497- 9507.
125. Zhigljavsky A, Hassan H, Heravi S. Forecasting European industrial production with singular spectrum analysis. *International Journal of Forecasting*, 2009: 103-118.